European Journal of Operational Research xxx (xxxx) xxx

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Interfaces with Other Disciplines

# The polynomial robust knapsack problem

Alessandro Baldo[a], Matteo Boffa[b], Lorenzo Cascioli[a], Edoardo Fadda[a,c,*], Chiara Lanza[a], Arianna Ravera[a]

[a] ISIRES, Torino, Italy
[b] Department of Electronics and Telecommunications, Politecnico di Torino, Torino, Italy
[c] Department of Mathematical Sciences "Giuseppe Luigi Lagrange", Politecnico di Torino, Torino, Italy

ARTICLE INFO

*Article history:*
Received 8 March 2021
Accepted 13 June 2022
Available online xxx

*Keywords:*
Heuristics
Robust knapsack problem
Genetic algorithm
Machine learning

ABSTRACT

This paper introduces a new optimization problem, namely the Polynomial Robust Knapsack Problem. It generalises the Robust Knapsack formulation to encompass possible relations between subsets of items having every possible cardinality. This allows to better describe the utility function for the decision maker, at the price of increasing the complexity of the problem. Thus, in order to solve realistic instances in a reasonable amount of time, two heuristics are proposed. The first one applies machine learning techniques in order to quickly select the majority of the items, while the second makes use of genetic algorithms to solve the problem. A set of simulation examples is finally presented to show the effectiveness of the proposed approaches.

## 1. Introduction

Our lives are the result of decisions; most of them are taken on the basis of intuition and common sense. More complicated decisions require, instead, a more systematic approach, and the adoption of proper methodologies of decision support, especially if such complex problems require to take into consideration a plurality of points of view. An effective example is when the decision maker is asked to choose among many conflicting projects, which may create some further extra economic value in the form of synergies among them (e.g. the common activation of a metro station and a parking area nearby might produce a traffic decrease bigger than the sum of those estimated for each single building).

This work aims at correctly mapping these practical situations into a suitable formulation, representing the joint development of two sub-classes of the traditional Knapsack Problem (KP), namely the Robust Knapsack Problem and the Polynomial Knapsack Problem. We call this new problem Polynomial Robust Knapsack Problem (PRKP).

The KP is a well known combinatorial optimization problem which given a set of items, each one characterized by a weight and associated to a utility value, aims at selecting the subset of items maximizing the total utility, by respecting a budget constraint on the items' weight. The problem often arises in resource allocation, where the decision maker has to choose from a set of non-divisible projects or tasks under a fixed budget.

However, the basic formulation of the KP is limiting with respect to real application scenarios. Indeed, it only considers a utility reward measure from the singular item, without modelling dependencies of contributions common to several items.

In order to fill this gap, the Quadratic Knapsack Problem (QKP) was introduced, extending the problem formulation to model the impact of quadratic terms in the utility function. Indeed, given a set of items, each with a weight and a value, an extra-profit is considered if two items are selected in the solution.

The Polynomial Knapsack Problem (PKP) is a further generalization of the Quadratic Knapsack Problem up to any polynomial degree terms. In order to linearize the products of the binary variables of each degree, this problem needs an exponential number of variables and constraints; in the real setting, the decision maker cannot define all such contributions, but in this way a more flexible model is provided, so that a limited amount of multiple contributions brought by increasing numbers of items can be taken into account if necessary. To the present knowledge, and probably due to this enhanced complexity, it has never been specifically studied in the literature.

The framework inside which the two aforementioned problems are defined only deals with a deterministic scenario: the informative content about each item (i.e. its profit and cost) is indeed known a priori. However, in several settings this may not be the case, as uncertainty is present over some of the items' characteristics. The Robust Knapsack Problem (RKP) is one possible way to

* Corresponding author at: Department of Mathematical Sciences "Giuseppe Luigi Lagrange", Politecnico di Torino, Torino, Italy.

*E-mail addresses:* alessandro.baldo@isires.org (A. Baldo), matteo.boffa@polito.it (M. Boffa), lorenzo.cascioli@isires.org (L. Cascioli), edoardo.fadda@polito.it (E. Fadda), chiara.lanza@isires.org (C. Lanza), arianna.ravera@isires.org (A. Ravera).

https://doi.org/10.1016/j.ejor.2022.06.029
0377-2217/© 2022 Elsevier B.V. All rights reserved.

face this lack of information. It represents an extension of the KP where the weight of each item varies in a pre-defined interval. The maximum number of items whose cost varies is governed by a parameter $\Gamma$, which is supposed to have been statistically estimated. Intuitively, the robust solution considers the worst case scenario, occurring when all the selected items achieve their respective largest allowed weight. If on the one hand the RKP more dynamically models the Knapsack Problem, on the other hand its framework does not originally include more than the singular items' contributions.

Given that the robust and the polynomial KP formulations have been always treated separately, the goal of this paper is to mix the two problems to fill such gap in the literature. Among the several different interpretations given for the aforementioned problems, for this paper it was chosen to deal with a set of investments as 'items', whose cost can vary in a bounded interval. The presence of several items leads then to a variation in the overall utility computation, which will be called *synergy*: therefore, the robustness of the problem co-exists with the polynomial utility terms. In this way, the scenario is adequate to describe the decision-making process that a company (or a public decision maker) could face having to choose between different options of investments: the opening of several investments may generate synergies (at several different combinatory levels) leading to an extra-profit. The objective is thus to maximize the minimum possible profit, taking into account the uncertainties brought by the variability, and always respecting the budget limit.

It will be shown how the cost of passing from a Quadratic to a Polynomial definition of the problem may be justified for the sake of generalization. Furthermore, enclosing the problem in the Robust context represents an important final step to obtain an ultimately complete formulation of the Knapsack Problem, well suited to represent even complex real-life problems.

The paper is organized as follows. A literature review about the PKP and the RKP is presented in Section 2. Then, the mathematical derivation of the problem is given in Section 3. Section 3.1 shows in practise how the mathematical problem could be contextualised in real world applications. Two heuristic methods are proposed in Section 4. Both algorithms exploit the continuous relaxation of the problem and merge some contributions from (respectively) the Machine Learning and Genetic Algorithms domains. Consequently, with Section 5 the discussion dives more deeply into the analysis of the computational time required by the exact solver to solve the model, followed by a close comparison of the performances of the proposed heuristics, proving their effectiveness in complex scenarios. The repository containing the code and the described experiments is publicly available.[1] Finally, Section 6 concludes the paper and sketches some possible future research lines.

## 2. Review of the literature

As already mentioned, the goal of this work is to investigate the Polynomial Robust Knapsack Problem, a generalization of the Knapsack Problem which embraces two distinct aspects of such field, *robustness* and *polynomial synergies*. The research that has been conducted to develop this analysis has shown that no or very little attention has been attributed in the past to this formulation of the Knapsack Problem.

The two separated approaches, however, have been thoroughly explored. The RKP, especially, has attracted relevant interest in recent years and has been addressed by several researchers and academics. It has been firstly introduced and studied in Eilon (1987), where the authors highlight how the optimal solution of a classical KP can be very sensitive to parameters changes, and it is indeed possible that a small change in the budget invalidates the found solution (see also Ben-Tal & Nemirovski, 2000). Robust approaches have therefore been adopted in order to be immune, up to a certain point, from data uncertainty.

In general, robustness in the KP may be linked with costs of the knapsack items (as in the following formulation) but also with their profits (e.g. Talla & Leus, 2014).

In Bertsimas & Sim (2004), the authors provide a detailed historical overview on how different solutions were thought and refined with time, not only for the RKP problem but for treating robustness in general. Robustness is indeed a key aspect in several real-world modelling problems: for example, in Nasri, Abdelmoutalib, Imad, & Jamali (2020) a robust approach for solving a vehicle routing problem with time windows of uncertain service and travel times is presented. In Monaci, Pferschy, & Serafini (2013) there is a clear representation of the robustness concept, insights on the linear programming modelling of the problem and some hints at interesting heuristic methods. Among these, a Dynamic Programming approach is presented to look for an exact solution of the problem. This, however, cannot be extended to the case analyzed in this work, as the property for which an optimal solution is composed of many sub-optimal solutions is not valid anymore when synergies are introduced.

Another dynamic approach for the solution of the robust problem is anyway explored in Aissi, Bazgan, & Vanderpooten (2007).

For what concerns the Quadratic feature, this has been introduced in Gallo, Hammer, & Simeone (2009), where the problem is solved through the use of 'upper planes' to be applied on branch-and-bound techniques. Also Chaillou, Hansen, & Mahieu (2006) works with a branch-and-bound approach which tries to find better approximations of the upper bounds using Lagrangian relaxations. A great effort has also been made in Pisinger, Bo Rasmussen, & Sandvik (2007) where the authors propose reduction methods using both Lagrangian relaxation and decomposition schemes. Instead, in Pisinger (2007) there is a complete review of the literature for the Quadratic-Knapsack problem, together with some efficiency analysis.

Meta-heuristic approaches are proposed in Hammaer & Rader (1997) for the 0–1 problem dividing the problem in smaller subproblems, while Billionet & Calmels (1996) exploits the continuous relaxation, as it will be done in the two heuristics proposed at the end this work. This method is then reviewed and expanded in Billionet, Fay, & Soutif (1999), which improves Billionet & Calmels (1996) with a more performing upper-bound search. Furthermore, Tabu Search for the QKP is applied in Glover, Kochenberger, Alidaee, & Amini (2002) and Létocarta, Nagihb, & Plateaua (2012), where the authors apply Lagrangian relaxation and Lagrangian decomposition as re-optimization to accelerate the solution. The extension of this problem to the Polynomial dimension is yet another thing that has received too little attention and that this work aims at improving.

In the broad range of heuristics applied to Knapsack problems, those using Machine Learning concepts are worth to be mentioned. Among them, a relevant work is Mirshekarian & Sormaz (2018), where the authors use ML for Combinatorial Optimization Problems. Here, an agent-based framework is applied to model a genetic algorithm, which will also take part in the heuristics implementation. In Kern, Lu, & Vasko (2020) as well, the Multidimensional Knapsack Problem (MKP) is tackled with a metaheuristic that exploits Teaching-Learning Based Optimization (TLBO). Other interesting heuristics worth to be mentioned have been presented in Ojstersek, Brezocnik, & Buchmeister (2020) and Rao (2020). However, it is important to remember that ML is not the solution for all the problems: for instance, in Amini, Banitsas, & Cosmas (2016) ML and heuristic results are compared for a particular

[1] https://github.com/EdoF90/PolynomialKnapsackProblem.git.

problem, whose outcome better addresses the heuristic counterpart. More in general, a thorough preliminary analysis of the problem needs to be done: whenever a limited number of training data is available, ML is not a valid alternative. On the other hand, with a higher data availability and a greater complexity, ML is often one of the most powerful approaches.

## 3. The mathematical model

In this section, the mathematical model of the aforementioned problem is presented. Before proceeding to the model presentation, it is worth stating that, given a decision maker who has to choose a subset of items to select among a set of items $\mathcal{I} = \{0, 1, \dots, I\}$, the utility function $u(x_1, \dots, x_I) : \{0,1\}^I \to \mathbb{R}$ can be expressed as:

$$u(x_1, \dots, x_I) = \sum_{i=1}^{I} w_i x_i + \sum_{i=1}^{I} \sum_{j=i+1}^{I} w_{ij} x_i x_j + \dots + w_{1 \dots I} x_1 \dots x_I. \quad (1)$$

It is important to recall that while first order terms are related to the utility of the single item, higher order terms are related to the interaction between them. For this reason, these will be called *synergies*, and in particular $w_A,\ A \subseteq \mathcal{I},\ |A| > 1$ will be called synergy of level $n > 1$ if $|A| = n$. It is worth noting that $w_A \in \mathbb{R}$, thus it can be positive or negative: if the $w_A > 0$ it means that the cumulative effect of the items is super-additive (i.e. given $x_1$ and $x_2$, $u(x+y) \geq u(x) + u(y)$), otherwise is sub-additive (i.e. given $x_1$ and $x_2$, $u(x+y) \leq u(x) + u(y)$). For example, positive synergies can be used in order to guarantee that the utility takes advantage of diversification (no risk measure can be a linear function of $x_1, \dots, x_I$). Instead, negative synergies can be used in order to model items that jeopardize each other (more on this in Section 3.1).

Since Eq. (1) is an expansion, it is reasonable to assume that, in general, higher order coefficients are smaller than the lower ones. Moreover, it is worth noting that, if a decision maker wants to describe its utility by means of the approximation in Eq. (1), given $I$ items, $2^I$ coefficients have to be specified. Thus, this would turn out to be practically infeasible for a big $I$. Due to this observation, the majority of the coefficients in Eq. (1) will be assumed to be zero.

The mathematical model of the aforementioned problem is described by the following sets:

- the set of possible items $\mathcal{I}$ with cardinality $I$;
- the set of synergies $\mathcal{A} \subseteq 2^{\mathcal{I}}$.

The parameters used by the mathematical model are:

- $W$: the capacity of the knapsack;
- $\Gamma$: maximum number of items whose cost can change from the nominal value;
- $p_i$: utility associated to item $i$;
- $w_A$: variation of utility if all the items in $i \in \mathcal{A}$ are chosen;
- $c_i^a$: nominal cost of item $i$;
- $c_i^u$: upper cost of item $i$.

The variables considered are $x_i \in \{0,1\}\ \forall\ i\ \in\ \mathcal{I}$, they assume value 1 if item $i$ is chosen, 0 otherwise. The general model is

$$\max \sum_{i=1}^{I} p_i x_i + \sum_{A \in \mathcal{A}} w_A \prod_{i \in A} x_i - (\max_{c_i} \sum_{i=1}^{I} c_i x_i) \quad (2)$$

$$\text{s.t.}\ \max_{c_i} \sum_{i=1}^{I} c_i x_i \leq W \quad (3)$$

$$x_i \in \{0,1\}$$

Eq. (2) is the objective function: the first two terms represent the utility from the choice, while the third one is related to the costs, and it expresses the concept of *worst-case* with a sub-problem of maximization of the costs for the chosen set of items. The same term, therefore, is also present in Eq. (3), which enforces the capacity limit. To model the synergies, the product $\prod_{i \in A} x_i$ is used. In order to linearize this term, the case in which $|A| = \{i, j\}$ can be first considered. To linearize $x_i x_j$, a variable $z_{ij}$ is introduced, such that:

$$x_i + x_j \leq 1 + z_{ij} \qquad \text{if } w_{ij} < 0 \quad (4)$$

$$x_i \geq z_{ij} \qquad x_j \geq z_{ij} \qquad \text{if } w_{ij} > 0 \quad (5)$$

In fact, if $w_{ij} < 0$ the model will set the variable $z_{ij}$ to 0, thus Eq. (4) enforces $z_{ij}$ to be 1 if both $x_i$ and $x_j$ are 1. Instead, Eq. (5) is needed because if $w_{ij} > 0$, the model will set the variable $z_{ij}$ to 1, thus Eq. (5) forces it to be zero if $x_i = 0$ or $x_j = 0$. By generalising the same reasoning for the higher order terms, Model (2)–(3) can be rewritten as:

$$\max_{x} \sum_{i=1}^{I} p_i x_i + \sum_{A \in \mathcal{A}} w_A z_A - \max_{c_i} \sum_{i=1}^{I} c_i x_i \quad (6)$$

$$\text{s.t.}\ \max_{c_i} \sum_{i=1}^{I} c_i x_i \leq W \quad (7)$$

$$\sum_{i \in A} x_i \leq |A| - 1 + z_A \qquad \forall\ A\ \in\ 2^{\mathcal{I}} \qquad \text{if } w_A < 0 \quad (8)$$

$$z_A \leq x_i \qquad \forall\ i\ \in\ A\ \in\ 2^{\mathcal{I}} \qquad \text{if}\ \ w_A > 0 \quad (9)$$
$$x_i\ \in\ \{0,1\}\ \forall\ i\ \in \mathcal{I}, \qquad z_A\ \in \{0,1\}\ \forall\ A \in\ 2^{\mathcal{I}}.$$

The maximization terms in problem (6)–(9) would be quite easy to be solved, if there was no assumption related to the parameter $\Gamma$: in fact, the maximization with respect to $c_i$ would boil down to $c_i = c_i^u$ for all the items, thus reducing the problem to a standard Knapsack Problem. Following the approach of Bertsimas & Sim (2004), however, it is assumed that no more than $\Gamma$ items change their prices with respect to their nominal values (and towards their upper values). The two terms containing the maximization with respect to $c_i$ (i.e., $\max_{c_i} \sum_{i=1}^{I} c_i x_i$) can be treated with the aid of *duality*. For the sake of simplicity and without loss of generality, consider first the capacity constraints:

$$\max_{c_i} \sum_{i=1}^{I} c_i x_i \leq W. \quad (10)$$

By using the definition of $c_i^a$ and $c_i^u$, Eq. (10) can be written as

$$\sum_{i=1}^{I} c_i^a x_i + \max_{\hat{c}_i} \sum_{i=1}^{I} \hat{c}_i x_i \leq W \quad (11)$$

where $\hat{c}_i \in [0, c_i^u - c_i^a]\ \ \forall\ \ i \in \mathcal{I}$ represent the deviations from the nominal costs introduced in the worst-case scenario. The focus for the maximization is in practice shifted towards the last part of the expression, which can be modelled as follows:

$$\max \qquad \sum_{i=1}^{I} [\hat{c}_i x_i] y_i \quad (12)$$

$$\text{s.t.}\ \sum_{i=1}^{I} y_i \leq \Gamma \quad (13)$$

$$y_i \in \{0,1\} \quad \forall i. \quad (14)$$

This brings to a sub-problem where the $\hat{c}_i$ and the $x_i$ are the coefficients for the introduced variables $y_i$. Practically, once an assignment has been done for the $x_i$, this sub-problem 'chooses' the worst possible $y_i$ to be set to 1 (at most $\Gamma$), taking at upper cost those items that maximize the total cost for the chosen set of items. The $y_i$ variable assumes value 1 iff the cost $\hat{c}_i = c_i^u - c_i^a$, i.e., it is set to the maximum of its deviation. It is important to notice that the problem can be relaxed considering the variable $y_i \in [0, 1]$ without changing the optimal solution. Thus, *duality* theory can be used to solve this problem.

Following the well-known dual transformation, indeed, the previous formulation can be re-written as follows:

$$\min \quad \Gamma\rho + \sum_{i=1}^{I} \pi_i \tag{15}$$

$$\text{s.t.} \quad \rho + \pi_i \geq \hat{c}_i x_i \qquad \forall i \in \mathcal{I} \tag{16}$$

$$\rho \geq 0, \quad \pi_i \geq 0 \quad \forall i \in \mathcal{I} \tag{17}$$

For what concerns the new dual variables, it has been chosen, in order to keep a notation equivalent to that used in Monaci et al. (2013), to build the vector $\lambda \in \mathbf{R}^{\mathbf{N+1,1}}$ of the Lagrangian multipliers as:

$$\lambda = [\rho, \pi_1, \ldots, \pi_i, \ldots, \pi_I]^T. \tag{18}$$

Specifically, $\rho$ is the dual variable associated to constraints Eq. (13) and $\pi_1, \ldots, \pi_I$ are the dual variables associated to constraints $y_i \leq 1$.

Therefore, Eq. (10) can be reformulated as

$$\sum_{i=1}^{I} c_i^a x_i + \min\ (\Gamma\rho + \sum_{i=1}^{I} \pi_i) \leq W, \qquad (16), \qquad (17). \tag{19}$$

The *min* is actually no more necessary, as the nature of the main problem, which aims at maximizing the value of the utility, will necessarily bring the expression of the costs to its minimum possible value. Deleting the minimum operator, the feasible space is expanded, but the final result will remain the same. An equivalent reasoning can be performed for the term $\max_{c_i} \sum_{i=1}^{I} c_i x_i$ in the objective function, and it has been verified that the same $\rho$ and $\pi_i$ variables may be uniquely adopted for both the cases, so that in practice each of the secondary maximization problems $\max_{c_i} \sum_i c_i x_i$ can be simply substituted with:

$$\sum_{i=1}^{I} c_i^a x_i + (\Gamma\rho + \sum_{i=1}^{I} \pi_i), \tag{20}$$

introducing in the model the previously defined dual constraints and adding Eqs. (16) and (17) in the constraints.

The final formulation of the model for the PRKP problem is therefore the following:

$$\max \sum_{i=1}^{I} p_i x_i + \sum_{A\in\mathcal{A}} w_A z_A - \sum_{i=1}^{I} c_i^a x_i - (\Gamma\rho + \sum_{i=1}^{I} \pi_i) \tag{21}$$

$$\text{s.t.} \quad \sum_{i=1}^{I} c_i^a x_i + \Gamma\rho + \sum_{i=1}^{I} \pi_i \leq W \tag{22}$$

$$\rho + \pi_i \geq (c_i^u - c_i^a) x_i \qquad \forall i \ \in \ \mathcal{I} \tag{23}$$

$$\sum_{i\in A} x_i \leq |A| - 1 + z_A \qquad \forall A \in 2^{\mathcal{I}} \qquad \text{s.t.} \quad w_A < 0 \tag{24}$$

$$z_A \leq x_i \qquad \forall\ i \in A \in 2^{\mathcal{I}} \qquad \text{s.t.} \quad w_A > 0 \tag{25}$$

$$x_i \in \{0, 1\} \quad \forall i \in \mathcal{I}, \qquad z_A \in \{0, 1\} \quad \forall A \in 2^{\mathcal{I}} \tag{26}$$

$$\rho \geq 0, \qquad \pi_i \geq 0 \quad \forall i \in \mathcal{I} \tag{27}$$

The constraints are respectively related to the maximization of the earning in the worst case, the knapsack capacity, the dual constraints used to linearize Eq. (3), the synergies activation and the variables domains. This kind of model implies that among the final set of items, the ones whose cost will assume the upper value are those for which such increase with respect to the nominal cost is larger.

### 3.1. Practical applications

In this brief section, we provide more clarity on the treated scenario, by discussing real world applications of Model (21)–(27). We consider two use cases.

In the first one, a public decision maker has to choose a subset of projects from a pool to improve the living condition of the population. On one side, its choices need to be robust to uncertainty: in other words, even when a bad scenario arises, the project must remain feasible (i.e., within the planned budget). On the other one, the decision maker is aware that several projects may have positive and negative synergies. For example, installing a set of electric charging stations in a parking area and refurbishing a near building for institutional purposes provide to the population more utility than just the sum of the two projects, meaning that the concurrent activation of both investments brings an additional gain.

In the second example, a decision maker, which we can think as an hedge fund, has to do strategic investments in a subset of macro-sectors while having access to relevant statistics about them (e.g. investments' costs, returns, volatilities, etc.).

These statistics can be used to define an instance of Model (21)–(27) in which:

- the utility $p_i$ is the expected future price;
- the nominal cost $c_i^a$ is the asset's market price;
- the upper cost $c_i^u$ is an upper bound on the asset's price (e.g. the price majored considering its historical volatility);
- the synergies keep into account cross-correlations across assets, with positive synergies associated to negatively correlated assets. Thus, the model will favour diversification, reducing risk.

Therefore, the model's solution provides the set of investments with the highest expected income (fitting into the budget at disposal), having taken into consideration all the interplays between themselves and the possible variability of their costs.

In conclusion, we showed some real world applications of Model (21)–(27), but several others could be considered. It is worth to mention that the aforementioned use cases are reported for illustrative purposes. Since not having real data at disposal would result in defining arbitrary instances of questionable quality, in Section 5 we define a general way to define random synthetic instances.

## 4. Heuristics

In this section, two heuristic methods for the solution of the presented problem are illustrated. The first one relies on the application of Machine Learning (therefore named **ML Heuristic**) concepts while the second relies on an adaptation of Genetic Algorithm (therefore named **Genetic Heuristic**).

### 4.1. ML heuristic

Machine Learning is a branch of Artificial Intelligence (AI) which aims at analysing and interpreting patterns and structures among data. To reach this goal a set of known input and output data (called training set) are fed to the ML algorithm which learns how to predict the output for a new, unseen input. For this specific problem, the task is *binary classification*, as the interest is on whether or not a variable will be set to 1 (i.e., the corresponding item is inserted in the knapsack).

More specifically, the basic idea is to exploit the power of a ML algorithm, which assigns to each binary variable of the problem the probability to assume value 0 or 1 in the optimal solution. From these estimated probabilities, the variables whose result is most certain are fixed. Contrarily, the most uncertain ones are further processed by the exact solver, which however benefits from the previous reduction of the search space. It follows that the complexity of the last run can be modified by simply changing the percentage of variables which are fixed by the classifier. This eventually provides a two-fold tool, which can either focus on short running times or on solution goodness. In the following, the variables considered by this approach will be the $x_i$ of Model (21)–(27). Since these variables are in a one-to-one correspondence with the items, we will interchange the terms “items” and “variables”.

In order to apply a ML algorithm, a predefined set of features to characterize each item must be defined. In the proposed approach, the features chosen to describe item $i$ of the considered problem are:

- the optimal value of the $\tilde{x}_i$ in the continuous relaxation of the problem (thus it will not be 0/1 anymore, but a value continuous in such interval);
- the profit associated to such item;
- its nominal cost divided by the budget (it represents the percentage of space occupied by the item, irrespective of the budget);
- its upper cost divided by the budget;
- the number of the positive synergies in which the item is included;
- the number of the negative synergies in which the item is included;

As mentioned, the label associated to each item is binary, indicating if the item is being left out or taken into the final solution.

The working principle behind this approach follows the general trend that items having high profit with low weights and taking part in a lot of positive synergies tend to be predicted in class 1 (thus taken into the knapsack). On the contrary, items characterized by heavy weights, low profits and leading to many negative synergies will hardly ever be considered as part of the final solution (then they are classified in class 0). With this piece of work, therefore, the search space for the discrete run is restricted to those items whose features are not descriptive enough of their goodness for the problem, and this consistently speeds up the computational time without heavily impacting the final result.

The general pseudocode of the heuristic is reported in Algorithm 1.

**Algorithm 1** ML heuristic.

```
1: solve PRKP continuous relaxation
2: prepare feature vector for each item
3: load previously trained classifier
4: predict class probabilities for each item
5: sort the class probabilities in descending order (most probable
   at the beginning)
6: classify the items with the highest probability to their predicted
   class
7: solve PRKP by fixing the variables related to the classified items
8: return solution
```

The second run of the solver, performed with the discrete problem and more computationally expensive, is therefore launched with a limited number of items needing to be discriminated; these are the remaining ones for which ML predictions are less reliable. The result of this execution is taken as final solution of the heuristic.

Clearly, some studies and tuning are needed for the refinement of the heuristic. For what concerns the classifier, a very large training set is created by generating 800 instances of the problem. For every variable $x_i$ of each instance, the feature vector is properly built and its labels are extracted after solving the problem through an exact solver. Then, many different algorithms for binary classification have been experimented on this set; namely, the considered ones were KNN, SVM, MultiLayer Perceptron, AdaBoost and Random Forest. The latter was chosen given the higher performances. A more precise detail on the classifier choice has been skipped as it is somehow independent on the peculiar heuristic procedure; in any case, accuracy results for the tested classifiers can be easily retrieved running the provided source code.

Finally, the main parameter of the proposed heuristic needing to be tuned is the percentage of variables to be fixed before running the exact solver. This is a core aspect of the heuristic, heavily influencing its performances, both in terms of optimality gap and execution times. Indeed, high values of this threshold would heavily reduce running times, but would lead to bad and possibly infeasible solutions. In fact, since the classifier does not consider the problem constraints but just the items characteristics, it might overcome the knapsack capacity leading to infeasible solutions that have to be adjusted. Conversely, too low values would increase the workload of the exact solver, thus requiring too much time for the solution computation (or leading to out of memory errors in the worst case). As a consequence, the role of the heuristic would be strongly harnessed.

Therefore, the fine-tuning procedure has been included into a Cross-Validation process, testing ranges from 75% to 100% of fixed items. Such analysis has been evaluated according to three key indicators: *(i)* the ratio between the running time of the heuristic and of the solver; *(ii)* the final solutions’ average optimality gap and its standard deviation; *(iii)* the generalization performance across different sets of instances, both in terms of types and dimensions (measured by the number of infeasible solutions obtained). As a result of the experimental campaign, the choice to fix 85% of the items has proven to be the best one under each of the aforementioned criteria, leading to no infeasible solutions.

It is worth noting that Cross-Validation as well as local search methods may be adopted to decide the percentage of variables to be fixed when applying the heuristic to other problems. In conclusion, it is important to notice that in real life problems it is likely that the sizes of data instances will be restricted to specific ranges, therefore enabling the possibility to adopt an even more tailored threshold.

## 4.2. Genetic heuristic

Genetic Heuristic is based on Genetic Algorithms (GA) which are known to offer large scalability on data, by maintaining a low overall complexity (see Vose, 1998).

The main structure is composed of an initial setting of the parameters followed by an iterative method composed of basic simple actions.

The initialization step creates a population of the so called *chromosomes*. As in the genetics field, each chromosome is a unique subset of attributes, where each attribute (i.e. a gene) is, in turn, a fundamental unique unit for the entire process.

After the creation step, the method iteratively computes an evolution of the population, passing through a *parents selection* step, where the best individuals of the population are selected according to a common criterium: the *fitness score*. Generally, the fitness score is chosen such that it maps the expression of the objective function.

Consequently to this operation, the algorithm provides for the evolution of the initial population through the *crossover* and *mutation* procedures. The former allows to renovate the set of chromosomes by emulating the real genetic crossover procedure. Therefore, from two individuals, two *siblings* are created by determining a splitting point in the two parents. In this way every new chromosome's attribute is derived by one of the two parents. The mutation is instead an operation which aims at increasing the randomness in the population, involving only a subset of chromosomes and transforming a gene (i.e. an attribute) in a different, unpredictable one.

Given the overall structure, a GA can have multiple possible stopping criteria, like a no-improvement policy, a maximum number of iterations or a target value for the convergence.

### 4.2.1. The implementation for the polynomial robust knapsack

The final version of the Genetic Heuristic implements the aforementioned setup in a more elaborated scenario. Indeed, with the intent of giving a sub-optimal initialization of the population, the genetic characterization takes advantage of the continuous relaxation of the PRKP. Its minor complexity allows to efficiently direct the initialization of the Genetic Algorithm, which would surely represent a time-consuming procedure in the economy of the heuristic. The solution of the continuous relaxation is interpreted under a probabilistic assumption; in other words the continuous decision variable is interpreted as the probability that the associated item is equal to one. This fosters the presence among the selected items of those whose continuous value is close to 1. In this way, the creation of the population results in a light computational method, relying on the simpler nature of the related continuous problem and the efficiency of an exact solver. Finally, this scheme can be classified under the category of the Matheuristics.

Diving more deeply into the genetic phase, the first remarkable parameter to decide is about the correct dimension of the population. Indeed, although a large and well diversified population would lead to decrease the gap on the result between the exact solution and the solution provided by the heuristic, the interest is to maintain a net gap in computational times with respect to the exact solver. Doing so allows to obtain a better scalable method, able to fast manage large instances, without losing in precision in terms of the correct objective function. The conducted experiments proved that the correct trade-off is setting the initial population size to about 70 individuals.

Each individual (i.e. chromosome) is conceived as a simple binary string of length $I$, where the items set at '1' result belonging to the solution. A chromosome is thus built by taking each item with a probability equal to $\tilde{x}_i$. In other words, by calling $\rho_i$ the realization of a uniform random variable between 0 and 1, we have that

$$\begin{cases} \text{Acceptance} & \text{if } \rho_i \le \tilde{x}_i \\ \text{Rejection} & \text{otherwise} \end{cases} \tag{28}$$

Even though this method suitably works in the vast majority of the cases, some exceptions make the heuristic struggle in achieving the correct creation of the population. Indeed, due to this probabilistic initialization, some instances highlight the drawback of completely relying on the continuous relaxation of the problem. In those cases, indeed, the population results being filled with only infeasible individuals under the discrete problem constraints, directing the entire algorithm far from the space of feasible solutions. In order to cope with this limit, it is necessary to introduce a new penalizing hyper-parameter $\epsilon$ for the algorithm, only activated on a small bunch of individuals. This operation allows to guarantee that, in such cases, the creation of the population leads to a minimal percentage of feasible individuals, correctly constraining these exceptions. For sake of brevity, the tuning procedure of this new parameter is not reported in the hereby description, but a consistent framework is to set $\epsilon = 0.03$ on the 40% of the initial population. Thus, Eq. (28), is updated to the following expression:

$$\begin{cases} \text{Acceptance} & \text{if } \rho \le \tilde{x}_i - \epsilon \\ \text{Rejection} & \text{otherwise.} \end{cases} \tag{29}$$

The resulting population from the first step then represents the first generation of chromosomes for the consequent iterative procedure.

The *parents selection* consists of evaluating the best individuals according to their *fitness score*. Such measure, as described before, is defined according to the objective function formulation, by evaluating profits, costs and polynomial gains determined by the genes in the chromosome, as well as by checking the feasibility of the solution according to the budget constraint.

This selection criterium allows to define an implicit termination criteria. Indeed, at each iteration $i$, the parents selection operates a truncation of the population, reducing it to a length of $\zeta_i$, corresponding to the best individuals. In this way, the end of the algorithm coincides with a population of length 1 (i.e. the only remaining individual is no more able to make the population evolve anymore). The number of individual of the population evolves according to:

$$\zeta_i = \frac{70}{2^i}, \qquad i = 0, 1, 2, .6. \tag{30}$$

After the parents have been selected, the core of the Genetic Algorithms provides for a simple and efficient method to simulate the evolution of the population: the *crossover*. For each couple of chromosomes (*parents*), a random crossover point is extracted, and a new couple of chromosomes is added to the population. It is evident, that each one of the siblings inherits some attributes from both the parents. The effectiveness of this step is the strength of the heuristic, allowing a large scale random search in the space of the solutions.

Ultimately, the *mutation* step is a needed passage, which randomly flips one item in the chromosome, either inserting or removing it.

The general pseudocode of the heuristic is reported in Algorithm 2.

**Algorithm 2** Genetic heuristic.

```
1: solve polynomial knapsack in continuous relaxation
2: create population
3: parent selection
4: while len(population) != 1 do
5:     crossover
6:     mutation
7:     parent selection
8: return solution
```

## 5. Computational experiments

In this Section, the results of the computational experiments are reported. The flow is divided as follows. First, Section 5.1 presents the methodological framework used for the computational experiments. Subsequently, Section 5.2 shows the computational time needed for exactly solving the mathematical model. Having done so, Section 5.3.1 presents results for ML Heuristic and Section 5.3.2 reports results for Genetic heuristic, with Section 5.3.3 comparing the results of the two. Finally, Section 5.3.4 presents an additional analysis on the two sub-problems, to prove the general validity of the adopted methods.

### 5.1. Instance generation

In this section we describe how the different instances used for the following tests have been generated, and how the model (21)–(27) behaves with respect to different instance characteristics. Due to the novelty of the problem, there are no references to be taken as a benchmark. When possible, inspiration has been taken from Gallo et al. (2009) and Taniguchi, Yamada, & Kataoka (2008); these describe how the standard instances for the Quadratic and Robust problems are created. Anyway, analysing a new problem, some new elements are of course added with respect to the cited references, and therefore, for eventual interests and future comparisons with other works, the specific instances built for this study are freely provided[2].

The trials are performed by considering input data of different sizes $I$; these range between 100 and 1500. According to Gallo et al. (2009), the nominal costs $c_i^n$ are uniform randomly drawn numbers between 1 and 50. The upper costs, on the other hand, refer to the Taniguchi et al. (2008)'s example: a positive random parameter $d \in \{0.3, 0.6, 0.9\}$ is introduced, and the augmented cost is obtained as $c_i^u = c_i^n(1+d)$. Both articles agree on randomly drawing the profits $p_i$ from the set [1,100].

In Gallo et al. (2009), the authors provide a modelling for positive synergies; anyway, to achieve a more realistic scenario in which both increments and decrements might occur considering joint investments, the range was enlarged considering also negative values. Furthermore, the bounds are weighted according to the synergies cardinality: this seems reasonable, since the perceived utility assigned to smaller sets of items is generally greater than the one for larger groups. In practice, $w_A$ is drawn from a uniform with support in $[[-100/|A|; 100/|A|]]$.

A last reference from the articles is about the capacity $W$; this is obtained as $\frac{\sum_i c_i^n}{m}$, with $m$ randomly extracted in the set $\{2, 3, 4\}$.

Referring again to the synergies, their numerosity is set based on the size of the dataset: being $k$ the synergies' degree, for more than 1000 items they are exponentially distributed $(I/2^{k-1})$; for a number of items between 300 and 1000 they are smoothed according to the squared root of the former exponential trend $(I/2^{\sqrt{k-1}})$; for a lower numerosity, their number decreases linearly as $(I/(k-1))$.

With this scaling-policy, the intention is to stress the exact model both in terms of synergies' density and dataset magnitude. The non-affordable times that the latter requires for the computations and, even more importantly, the excessive complexity leading to missing results, testify the real benefits brought by the heuristics.

For the choice of the parameter $\Gamma$, Fig. 1 shows some interesting empirical results which are indeed helpful for a better understanding of the problem. With a fixed capacity, the running time of the exact solver as function of $\Gamma$ follows a Gamma like shape where the maximum is generally disposed on the left half of the distribution; this helps understanding that the most time-consuming configuration for $\Gamma$ refers to a value between 20% to 60% of the total numerosity. Setting a lower or greater value would tend to the best or worst case in terms of cost deviations but in any case reducing the overall complexity to be closer to a standard KP. These are not general rules, since the model is affected by many other parameters; anyway, for the described setting this range is well suited for appreciate the complexity of the proposed models and it is therefore adopted for the next experiments.

[2] https://github.com/EdoF90/PolynomialKnapsackProblem.git.

With a more enlarged analysis, in Fig. 1a it is possible to notice how the influence on computational times is not only dictated by the proportion of $\Gamma$ with respect to the total number, but also by the density of the existing synergies between items. An higher number of interactions between elements thus is reflected in increased computational times. Finally, Fig. 1b underlines the strong relation between the numerosity of the dataset and the execution times of the model. Specifically, with the same condition on the density of synergies, doubling the numerosity computational times do not increase linearly, highlighting the exponential nature of the PRKP.

### 5.2. Model running times

This brief paragraph is going to show how the computational time required to optimally solve the PRKP rapidly increases with respect to the increment in the number of considered items.

Results have been computed by means of a Dual-Core Intel Core i5 CPU 2,7 gigahertz and 8 GB RAM machine. Problem instances are solved to optimality by the commercial solver Gurobi V9.1.

In order to extract some benchmark execution times by the exact solution of Model (21)–(27), several runs of the model were executed for a large set of randomly generated instances, built as previously explained and grouped into 11 categories with growing number of items. Running times for each category are reported in Fig. 2.

As it can be seen, time complexity is exponentially increasing with the items numerosity (and indeed basically linear in logarithmic scale): these graphs stop at numerosity equal to 700, but since the goal is to go well beyond that, it is clear that alternative methods are needed. This result confirms the general theory of complexity and justifies the development of heuristics to solve the problem.

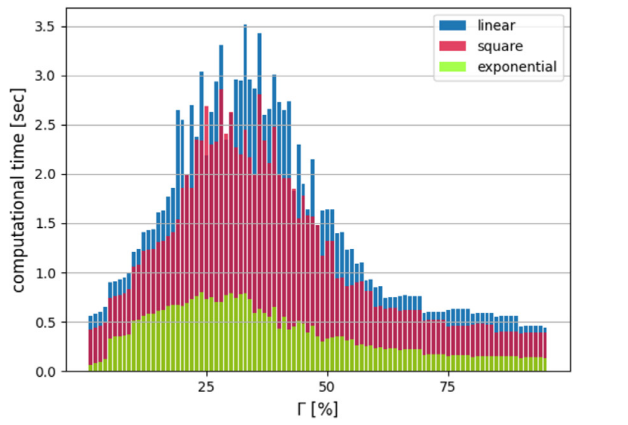
### 5.3. Heuristics

In the following, the heuristics performance and their effectiveness are measured. Since the exact solver computational time grows exponentially with the numbers of items, we set the time limit of the exact solver executions to 30 minutes. The analyses showed on the figures keep track of the number of times in which this bound has been achieved by inserting a labels on top of each boxplot. All the tests are run on a total amount of 2000 instances, equally divided into 8 categories of the same cardinality. As already discussed in Section 5.1, these instances can be still divided in three sub-categories, according to how the synergies were treated.

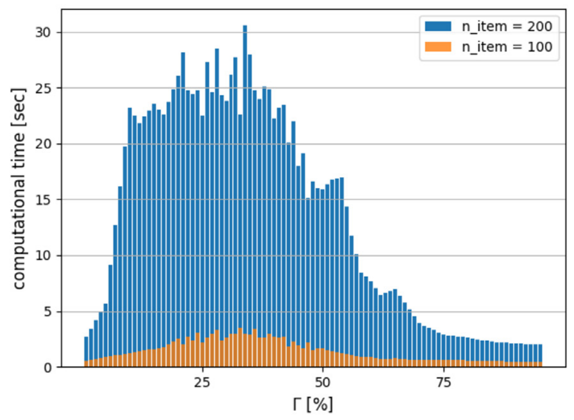
#### 5.3.1. ML heuristic

ML Heuristic seems to behave very well in terms of optimal solution compared to the model, as shown by Fig. 3. Indeed, despite the quite little percentage of items which are left as undecided before the final discrete run, the results do not seem to worsen, indicating that the choices taken after ML predictions are generally correct. The only cases in which the 5% gap margin is exceeded by some outliers are for instances of reduced dimensions: this may happen in smaller instances where a wrong choice on few items has great impact.

To be fair, it must be said that actually raw ML predictions on the class of each item are generally extremely precise, with an accuracy well above 90%. On the other side, though, this is not enough to exclusively rely on such predictions for the problem solution, as the nature of the task states that what must be very accurately maximized is the objective function value. In the objective function, also a few errors made on very 'heavy' items (in terms of contributions to the profit) can significantly alter the final result:



(a) Different distributions of synergies with 100 elements



(b) Linear distribution of synergies varying numerosity

**Fig. 1.** Exact solver computational time for different instance characteristics.

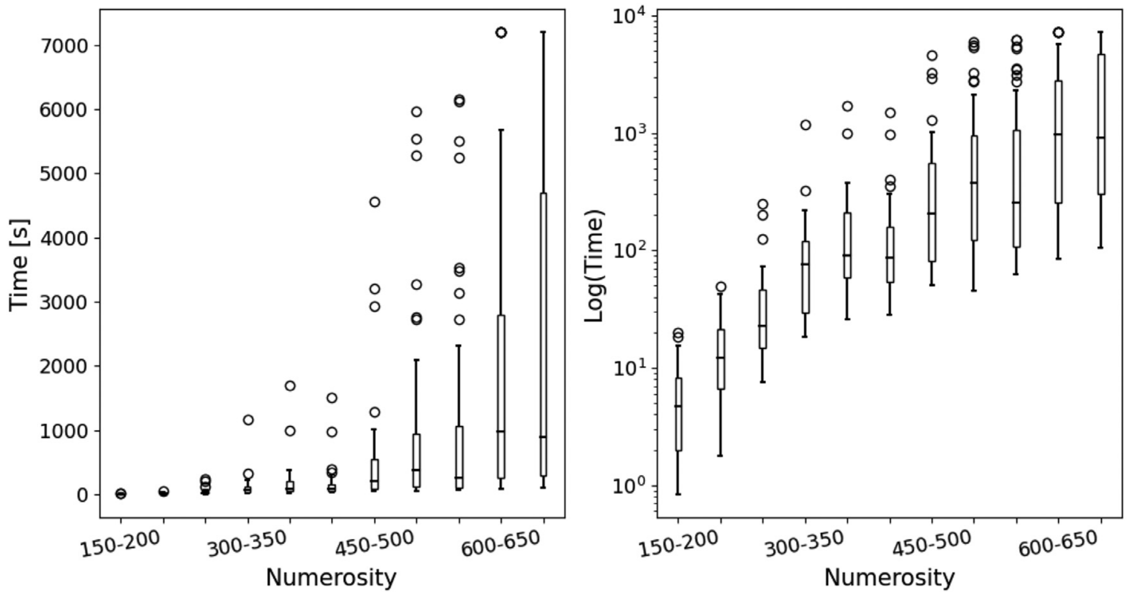


**Fig. 2.** Model running times.

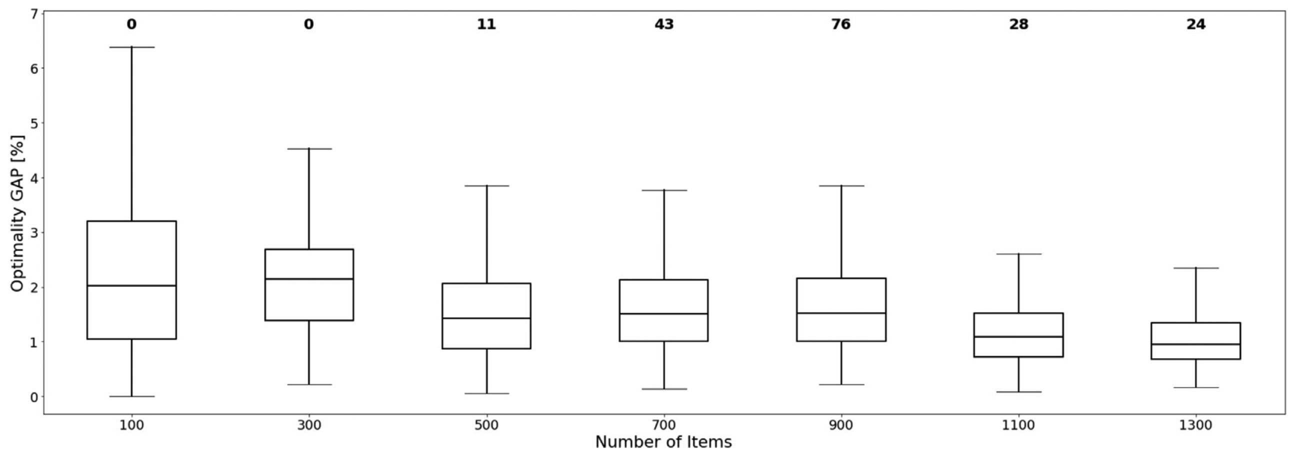


**Fig. 3.** Result: model vs. ML heuristic.

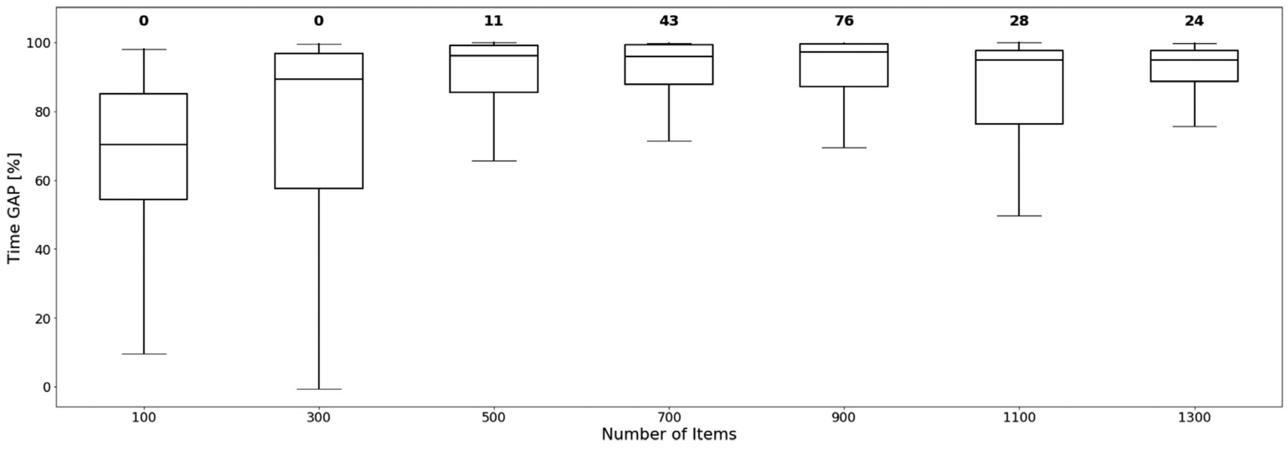


**Fig. 4.** Computational time: model vs. ML heuristic.

this is why ML can be used to greatly reduce the search space, but a discrete run of the solver is anyway needed.

As instance dimensions grow, this system is always able to find a solution acceptably close to the optimal one, or to the best solution found by the solver after 30 minutes of run, as mentioned before.

It has been here shown how the heuristic is able to greatly limit the gap with respect to the optimal solution: in almost every case which requires a long run of the model, ML Heuristic is able to provide a result which is usually very close to the desired one. That said, to see the fundamental gain brought by the heuristic, a quick look at Fig. 4 will be enough. The values reported here represent how faster the heuristic is with respect to the model. For relatively small instances, the two perform quite similarly as expected, but as soon as the model begins to struggle, time performances of ML Heuristic remain good. The item fixing before the discrete run allows to keep the execution time very low, as the other previous steps take a time which is usually negligible with respect to the discrete run time. Therefore it can be highlighted that from instances of 500 items onward, on average the heuristic keeps running in (roughly) 10% of the time needed by the model (see Fig. 4). In some cases, as recalled by the small labels on top of the boxes, the model has been stopped because its running time exceeds the 30 minutes threshold, so the results in the graphs are actually a lower bound on the real performance of the method.

In conclusion, we claim that the ML Heuristic has proven to be a very valuable tool to solve big instances of the Polynomial Robust Knapsack Problem, managing to obtain absolutely acceptable results with a very consistent reduction of the execution times.

### 5.3.2. Genetic heuristic

Genetic Heuristic is conceived as a method, able to generalize and adequately scale on a large variety of instances. The comparison with the exact solver performance in terms of gaps and computational times are shown in Figs. 5 and 6, respectively.

The proposed heuristic well behaves towards the model, being mostly (as, for each category, over the 75% of the cases prove) bounded below the optimality gap of 5%. A minor number of outliers determines an expected variability of this method, which can be intuitively attributed to the probabilistic nature of the algorithm.

Another aspect highlighting the capabilities of such a method is that the heuristic has also been tested with the constraint of a fixed length for the population of individuals. For this reason, the increase of the variability as the instances get larger can be considered statistically irrelevant.

The former considerations can be extended to the comparison over the computational times, where the heuristic is significantly upper-bounded by the threshold represented by the value 1 (i.e. when the heuristic method took the same computational time as the exact solver). In this scenario, the interest is mainly focused on the behaviour of the central and right-most part of the chart, since it is still likely that the exact solver and the heuristic, on quite small but very dense instances, could perform similarly. At the same time, where the instances are quite sparse, with still a limited numerosity (e.g. 1100 items), the exact algorithm still performs better in some cases, which are though not very meaningful for this scope.

On the other hand, under the computational speed point of view, the trend denoted by the instances with medium density is a good generalization of the average behaviour of the heuristic. Indeed, where the exact algorithm struggles to find an optimal solution (especially in those many examples where the time-limit was reached), the heuristic proves to be some orders of magnitude faster.

Furthermore, it is interesting to examine the behaviour in the right extreme of the chart, underlining the expected scalability of the Genetic Heuristic. In that region, as opposite, the exact solver shows some difficulties in managing large instances, despite the density of synergies falling in the best-case scenario.

Finally, Genetic Heuristic proves to be more reliable than the exact solver in several instances and it proves to be a valid counterpart of the exact algorithm, given its large scalability and adaptability in coping with the computational time limits of the Gurobi solver, especially when facing dense and/or large instances of the problem.

### 5.3.3. Comparison

In this section, a more precise comparison is shown with the aim to analyze how the two heuristics perform with respect to each other.

Table 1 presents a detailed quantification of the heuristics' results, both in terms of running time and optimality gap. The average run time and the average gap from the model solution are reported for both the heuristic methods, to allow also some comparisons between them. Moreover, as done in the previous plots, the instances of the problem are divided into groups of growing numerosity.

The Table shows once again how the two methods perform generally very well and do not heavily suffer when the dimensions of the problem increase. Particularly, ML Heuristic performs even better in the considered instances, as the mismatches with the exact solution, and so the optimality gap, do not scale like the increas-

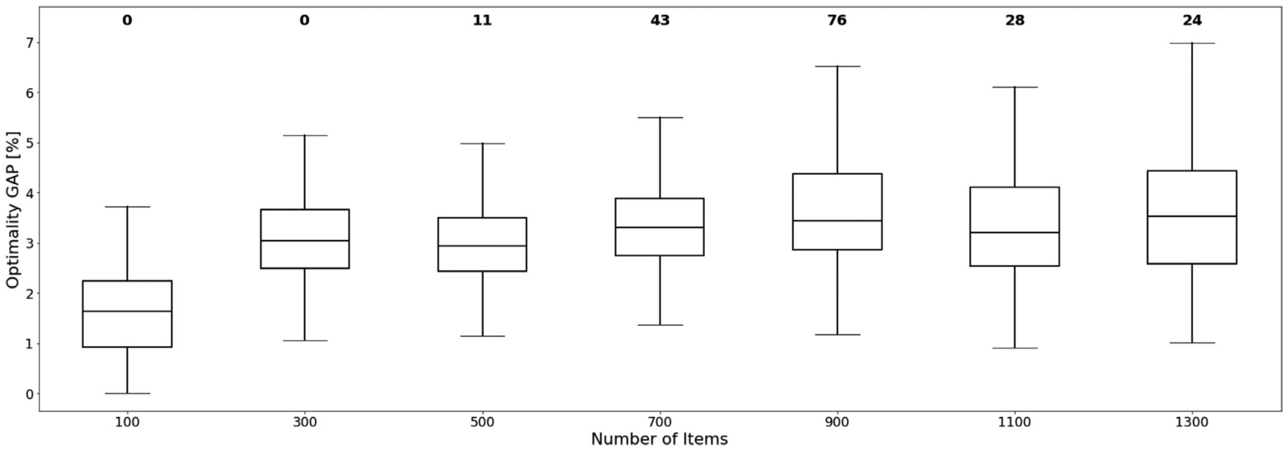


**Fig. 5.** Result: model vs. genetic heuristic.

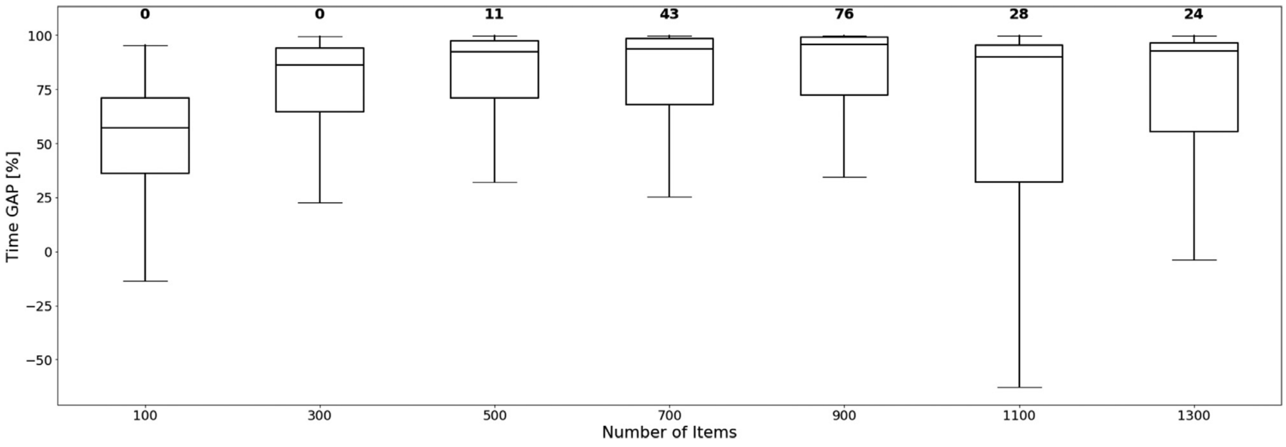


**Fig. 6.** Computational time: model vs. genetic heuristic.

**Table 1**
Mean [Stdev] for ML heuristic and genetic Heu run time and % gap.

| | | ML Heu | | Genetic Heu | |
|---|---|---|---|---|---|
| *n* items | syn type | run time [s] | optimality gap [%] | run time [s] | optimality gap [%] |
| 100 | L | 0.34 [0.14] | 2.32 [1.62] | 0.62 [0.43] | 1.72 [1.05] |
| 300 | L | 3.07 [0.74] | 2.25 [1.16] | 4.38 [2.42] | 3.06 [0.92] |
| 500 | S | 2.37 [0.61] | 1.53 [0.84] | 5.43 [3.22] | 3.00 [0.84] |
| 700 | S | 4.59 [1.39] | 1.62 [0.85] | 9.51 [6.01] | 3.35 [0.87] |
| 900 | S | 7.32 [2.18] | 1.64 [0.81] | 14.24 [10.57] | 3.65 [1.11] |
| 1100 | E | 3.51 [3.33] | 1.12 [0.55] | 6.50 [3.46] | 3.35 [1.13] |
| 1300 | E | 5.10 [4.58] | 1.06 [0.56] | 8.45 [5.49] | 3.56 [1.21] |
| 1500 | E | 7.10 [8.47] | 1.10 [0.49] | 10.03 [7.19] | 3.82 [1.30] |

**Table 2**
Mean [Stdev] for heuristics' applied on sub-problems (run time and % gap).

| | ML Heu | | Genetic Heu | |
|---|---|---|---|---|
| sub-problem | run time [s] | optimality gap [%] | run time [s] | optimality gap [%] |
| Robust | 1.49 [3.34] | 0.24 [0.23] | 7.40 [6.75] | 0.86 [0.67] |
| Polynomial | 0.78 [0.40] | 2.11 [0.72] | 13.78 [10.56] | 2.83 [0.72] |

ing number of items. This leads to a decreasing trend, in which the higher the numerosity, the lower the gap. Contrarily, Genetic Heuristic has higher margins, which however are always bounded within 5% on average.

### 5.3.4. *Heuristics on sub-problems*

As a last test, the generalization capabilities of the two heuristics have been evaluated on the sub-problems of the PRKP i.e. the RKP and the PKP. A subset of representative instances has been created (as described in Section 5.1), just putting to zero the synergies (for robust-only cases) or setting the upper costs equal to the nominal ones (for polynomial-only). The average results for instances with 300–1000 items are shown in Table 2. As expected, ML Heuristic (applied with the same trained classifier used above) behaves very well also in robust-only or polynomial-only instances. This was somehow expected, being the sub-problems alone much simpler than the complete one; nevertheless it was worth verifying that no overfitting or distorted modelling was occurring in the classifier. Genetic Heuristic as well shows overall acceptable (even if slightly worse) performances. Moreover, it is interesting to notice that the most challenging component between the two seems to be the polynomial one. As a final remark, the two sub-problems alone are extremely easier to solve, hence execution times are not really a remarkable issue in this case, even if heuristics still manage to improve.

## 6. Conclusions

The paper presented a new framework for the Knapsack Problem, merging the contributions of the Polynomial Knapsack Problem and the Robust Knapsack Problem. The novel formulation maps real-life problems where decision makers need to discriminate between an high number choices, whose impact can vary both in terms of costs and profits, also due to the existence of synergy terms among them. In order to effectively solve the problem, two heuristic methods are proposed. Both of them are based on the continuous relaxation of the problem. The heuristic exploiting Machine Learning incorporates the result of the relaxed problem as a pivotal feature to determine the correct classification of the item (taken or not inside the knapsack), while the Genetic Algorithm exploits such information under a probabilistic assumption, for the creation of the first population of individuals.

If on the one hand the latter is a widespread optimization method in the field, ML Heuristic opens a new perspective for further applications involving more complex and accurate models. Here the focus was on Random Forest Classifier, whose characteristics of an ensemble method ensured high scores in terms of accuracy under different test circumstances. However, the possibility to move the spotlight to more complex, highly-dimensional frameworks could be worth considering, in order to constrain even more the subsequent resolution of the discrete problem. In this sense, if many low-dimensional binary classifiers prove to be adequate to the situation (i.e. Decision Trees, Support Vector Machines, Logistic Regression, Deep Learning, etc.), testing the deep learning counterpart could still improve the settings, without needing further resolutions of the solver.

Furthermore, the flexibility of ML Heuristic allows to easily expand this framework to other binary and integer optimization problems, where the values assumed by the problem's variable could be thought as the result of a classification task. Finally, nowadays the progress in the Machine Learning field can ensure to scalably test a large variety of different techniques with high-level frameworks and libraries, making this kind of applications more accessible also for the optimization field.

## References

Aissi, H., Bazgan, C., & Vanderpooten, D. (2007). Approximation of min-max and min-max regret versionsof some combinatorial optimization problems. *European Journal of Operational Research, 179*(2), 281–290. https://doi.org/10.1016/j.ejor.2006.03.023.

Amini, A., Banitsas, K., & Cosmas, J. (2016). A comparison between heuristic and machine learning techniques in fall detection using kinect v2,. https://doi.org/10.1109/MeMeA.2016.7533763.

Ben-Tal, A., & Nemirovski, A. (2000). Robust solutions of linear programming problems contaminated with uncertain data. *Mathematical Programming volume, 88*(3), 411–424. https://doi.org/10.1007/PL00011380.

Bertsimas, D., & Sim, M. (2004). The price of robustness. *Informs, 52*(1), 35–53. https://doi.org/10.1287/opre.1030.0065.

Billionet, A., & Calmels, F. (1996). Linear programming for the 0–1 quadratic knapsack problem. *European Journal of Operational Research, 92*(2), 310–325. https://doi.org/10.1016/0377-2217(94)00229-0.

Billionet, A., Fay, A., & Soutif, E. (1999). A new upper bound for the 0–1 quadratic knapsack problem. *European Journal of Operational Research, 112*(3), 664–672. https://doi.org/10.1016/S0377-2217(97)00414-1.

Chaillou, P., Hansen, P., & Mahieu, H. (2006). Best network flow bounds for the quadratic knapsack problem. *Combinatorial Optimization, 1403*, 225–235. https://doi.org/10.1007/BFb0083467.

Eilon, S. (1987). Application of the knapsack model for budgeting. *Omega, 15*(6), 489–494. https://doi.org/10.1016/0305-0483(87)90006-5.

Gallo, G., Hammer, P. L., & Simeone, B. (2009). Quadratic knapsack problems contaminated with uncertain data. *Combinatorial Optimization, 12*, 132–149. https://doi.org/10.1007/BFb0120892.

Glover, F., Kochenberger, G., Alidaee, B., & Amini, M. (2002). Solving quadratic knapsack problems by reformulation and tabu search: Single constraint case, *Series on applied mathematics* (pp. 111–121). Combinatorial and Global Optimization. https://doi.org/10.1142/9789812778215_0008.

Hammaer, P., & Rader, D. J. (1997). Efficient methods for solving quadratic 0–1 knapsack problems. *INFOR: Information Systems and Operational Research, 35*(3), 170–182. https://doi.org/10.1080/03155986.1997.11732327.

Kern, Z., Lu, Y., & Vasko, F. (2020). An or practitioner's solution approach to the multidimensional knapsack problem. *International Journal of Industrial Engineering Computations*, 73–82. https://doi.org/10.5267/j.ijiec.2019.6.004.

Létocarta, L., Nagihb, A., & Plateaua, G. (2012). Reoptimization in Lagrangian methods for the 0–1 quadratic knapsack problem. *Computers and Operations Research, 39*(1), 12–18. https://doi.org/10.1016/j.cor.2010.10.027.

Mirshekarian, S., & Sormaz, D. (2018). Machine learning approaches to learning heuristics for combinatorial optimization problems. *Procedia Manufacturing, 17*(2), 102–109. https://doi.org/10.1016/j.promfg.2018.10.019.

Monaci, M., Pferschy, U., & Serafini, P. (2013). Exact solution of the robust knapsack problem. *Elsevier, 40*(11), 2625–2631. https://doi.org/10.1016/j.cor.2013.05.005.

Nasri, M., Abdelmoutalib, M., Imad, H., & Jamali, A. (2020). A robust approach for solving a vehicle routing problem with time windows with uncertain service and travel times. *International Journal of Industrial Engineering Computations*, 1–16. https://doi.org/10.5267/j.ijiec.2019.7.002.

Ojstersek, R., Brezocnik, M., & Buchmeister, B. (2020). Multi-objective optimization of production scheduling with evolutionary computation: A review. *International Journal of Industrial Engineering Computations, 11*(3), 359–376. https://doi.org/10.5267/j.ijiec.2020.1.003.

Pisinger, D. (2007). The quadratic knapsack problem-a survey. *Discrete Applied Mathematics, 155*(5), 623–648. https://doi.org/10.1016/j.dam.2006.08.007.

Pisinger, D., Bo Rasmussen, A., & Sandvik, R. (2007). Solution of large-sized quadratic knapsack problems through aggressive reduction. *Informs Journal on Computing, 19*(2), 280–290. https://doi.org/10.1287/ijoc.1050.0172.

Rao, R. (2020). Rao algorithms: Three metaphor-less simple algorithms for solving optimization problems. *International Journal of Industrial Engineering Computations, 11*(1), 107–130. https://doi.org/10.5267/j.ijiec.2019.6.002.

Talla, N. F., & Leus, R. (2014). Complexity results and exact algorithms for robust knapsack problems. *Journal of Optimization Theory and Applications, 161*(2), 533–552. https://doi.org/10.2139/ssrn.1967411.

Taniguchi, F., Yamada, T., & Kataoka, S. (2008). Heuristic and exact algorithms for the max–min optimization of the multi-scenario knapsack problem. *Computers and Operations Research, 35*(6), 2034–2048. https://doi.org/10.1016/j.cor.2006.10.002. Part Special Issue: OR Applications in the Military and in Counter-Terrorism.

Vose, M. D. (1998). *The simple genetic algorithm: Foundations and theory*. Cambridge, MA, USA: MIT Press.