An Experimental Analysis of three Computational Approaches for Minimizing Losses on Electricity Distribution

João Carlos. H. Moreira, Marcos Daniel. V. Baroni, Flávio M. Varejão,

***Abstract*—Electricity losses during distribution due to non-technical sources are among the major causes of profit loss for electricity distribution companies (EDCOs). In Brasil, part of that profit loss can be recovered through an increase in the energy bill. However, the maximum value of that increase is limited by the regulatory agency in the form of non-technical loss reduction goals. The optimization problem addressed in this work treats the loss reduction problem from the EDCOs point of view. In order to achieve the reduction goals established by the regulatory agency, EDCOs have several loss reduction actions which are allocated in multiyear loss reduction plans. These plans try to achieve the goals without exceeding some predefined budgets, always aiming to obtain the highest possible profit for the EDCO. This work approaches the problem of the plan's definition as a generalization of the knapsack problem. A formal model is defined as an integer programming problem, and it's hardness is analysed through computational experiments using a generic solver applied to a variety of artificial instances. Two heuristics are then proposed, the first based in a greedy approach and the second based on the Tabu Search metaheuristic, and applied to the problem. Finally, the three approaches are compared considering the quality of the solutions found.**

***Index Terms*—OR in energy, combinatorial optimization, meta-heuristics**

I. Introduction

ONE of the biggest problems Electricity Distribution Companies (EDCOs) face is that of electricity loss due to non-technical sources. Those *non-technical losses*, caused mainly by factors unrelated to the electricity transmission process (like frauds or broken power meters), represented about 13% of all electricity distributed in 2012, according to a regional Brazilian distribution company [1].

All that loss imposes a great profit reduction to the EDCOs, which motivates them to recover it by increasing the price of the distributed electricity. However, in order to prevent abuses and encourage EDCOs to improve their services, the brazilian regulatory agency of electricity (ANEEL) sets a threshold on the amount of non-technical loss that can be covered by increasing charges, which is known as *non-technical loss reducing goal*.

In practice, the existence of this goal implies that EDCOs cannot charge clients more than a fixed value per kilowatt-hour to attenuate those non-technical losses. The non-technical loss reducing goal (and indirectly the maximum kilowatt-hour charge) is established by ANEEL by studying the EDCO's profile and the historical non-technical losses behaviour at similar regions of the country. Additionally, the EDCOs are required to *reduce* the kilowatt-hour electricity charge if they manage to mitigate non-technical electricity loss above the established goal. This means that there is no gain to the EDCOs if they reduce the non-technical losses above the established goal.

Usually, the non-technical loss reducing goal is given for a period, which is usually three years, defining a *non-technical loss reducing curve*. If EDCOs don't meet the imposed non-technical loss reduction goal in a given year, their profit in that year is reduced by the corresponding value of the electricity under the goal. For example, suppose an EDCO with a historical non-technical electricity loss level of 15%. That means that considering only the non-technical losses, in order to sell 85 GW of electricity the EDCO must buy 100 GW from the electricity generation companies. If the regulatory agency establishes a hipotetical goal of 10%, the EDCO is authorised to make an increase in the electricity charge to cover 10 of the 15 GW lost, and must mitigate the other 5 GW, or assume the corresponding profit loss.

In order to fight non-technical electricity losses and reach the goal, the EDCOs define *Non-Technical Losses Reduction Plans* for a given period. Those plans consist of the non-technical loss reduction curve, a portfolio of *Non-technical Loss Reducing Actions*, *yearly budgets* for different *resources*, which will be spent on the execution of the actions and an *internal return rate*. A great variety of actions compose the portfolio, and each one of them has several distinct characteristics which must be considering during planning. One of them is the action's *market*, representing how many times the action can be executed during the whole plan. Another one is the *yearly market*, which is analogous to the market, but represents the amount of times an action can be executed during each year of the plan. Other important aspects are the action's *cost*, *recovered electricity*, *electricity value* and *dependency*. Firstly, each action consumes different portions of each budget, that is called the action's cost. Also, even though an action consumes the budgets only on the year it was executed, the electricity recovered by the action may be spread over the following years. Another important characteristic is that different actions may have a diferent values for each unit of electricity recovered, so in order to calculate the action's profit, the *electricity value* must be defined for each action. Finally, some actions may require other actions to be taken before them, so a *dependency* relation between those actions must be defined.

Research supported by Fundação de Amparo à Pesquisa do Espírito Santo.

This scenario introduces an interesting problem to the EDCOs: given an investment portfolio comprised of several non-technical loss reducing actions, a non-technical loss reduction curve and yearly budgets, what is the best possible allocation of actions that maximizes the return of the investment, that is, the allocation which yields the greatest profit for the EDCO? If the EDCO chooses to do nothing, it will not be able to charge for some portion of the distributed electricity, and since this usually costs more than performing the non-technical loss reducing actions, doing nothing is not the optimal course of action. On the other hand, reducing the loss to a greater extent than the imposed reduction curve is also not the optimal decision, since the over-reduced electricity loss does not increase the overall EDCO profit.

The current methodology used by EDCOs is to manually select the loss reducing actions in a heuristic trial-and-error fashion. As we shall present in this work, choosing the best non-technical loss reducing actions is a complex combinatorial problem, thus the current *de facto* procedure hardly finds an optimal solution of the problem. Since reducing non-technical losses is an important activity of EDCOs, this motivates the use of computational techniques to find the best, or at least good solutions to the problem.

In this paper, we formulate the problem previously presented formally as a generalization of the knapsack problem, a well known problem in Combinatorial Optimization. Due to the novelty of the formulation and some constraints on the availability of real instances of the problem, an instance generator is also presented, and we define a set of benchmark instances. Two heuristics are also proposed to solve the problem, one based on a greedy approach, named GALP, and the other on the Tabu Search metaheurisic, the TSLP. Experimental tests are then conducted, solving the instances with the CPLEX solver and the two proposed heuristics, and a comparison between those three approaches is made.

The remainder of the paper is organized as follows: Section II defines formally the problem of choosing non-technical loss reducing actions, and explain its relation to the knapsack problem. Section III introduces the approches used for solving the problem. Section IV introduces the instance generator used to create our set of benchmark instances. In Section V we present our experimental evaluation. Finally, in Section VI we make our concluding remarks.

II. Problem Definition

In this section we show the mathematical model defined to tackle the EDCO's problem previously introduced at section I. Firstly, section II-A details the mathematical model proposed to solve the problem. This section is followed by section II-B, where we show how the model defined relates to the available literature, more specificcally, we show that the model is a generalization of the well known knapsack problem, and discuss about the problem hardness.

A. Mathematical Model

In order to apply optimization methods to the problem, we must first define it formally. The model defined in this work considers that the only objective is to maximize the *Net Present Value* (NPV) of the investment made, that is, maximize the financial return of the investments, for a plan of M years, given:

- The *yearly budgets* $o_{i,l}$, for a set of L resources, $1 \leq i \leq M$, $1 \leq l \leq L$;
- the *yearly reduction goals* g_i, $1 \leq i \leq M$, representing the amount of electricity loss that must be reduced on the $i-th$ year;
- the *internal return rate* r, which represents the annual depreciation of the investment. This rate is constant for all years.

The investment to be made consists of choosing a subset of actions from a portfolio with N actions. Each $j-th$ action from this portfolio, $1 \leq j \leq N$, has a set of characteristics:

- The *elecricity value* v_j, representing the value of each unit of electricity recovered by the $j-th$ action;
- m_j, the *market* of the action, i.e. how many times action j can be executed during the whole plan;
- $u_{j,i}$, the *yearly market* of the action, or how many times action j can be executed on the plan's $i-th$ year;
- $c_{j,l}$, how much each execution of action j consumes from resource l, in other words, the *cost* of the action;
- $e_{j,k}$, the *electricity recovered* by action j on the $k-th$ year after its execution, given in the form of an electricity recovery curve since an action can recover electricity, i.e. avoid electricity loss, on the year it was executed and on the following years;
- a set D_j of pairs $(d, Q_{j,d})$ representing the *dependencies* of action j. For each execution of action j, each action $d \in D_j$ must be executed previously an amount of time defined by $Q_{j,d} \in \mathbb{R}^+$.

The objective is to find a solution $\bar{x}$, in other words, a set of values for the decision variables $x_{j,i}$, $\forall i, j$, $x_{j,i} \in \mathbb{N}$ that maximizes the NPV. This solution represents how many times action j will be executed on the $i-th$ year of the plan. In order to present the NPV equation and consequently the problem's objective function, three auxiliary equations must be defined first:

- The first of them, equation 1, represents the electricity loss reduction caused on the $i-th$ year by action j executed on the $k-th$ year of the plan. In other words, it calculates how much electricity loss the execution of action j on the year k will avoid at year i:

$$R_{i,j,k}(\bar{x}) = x_{j,k} \cdot e_{j,i-k+1} \tag{1}$$

- The second, equation 2, represents the total yearly profit V_i, which is the sum of all energy recovered on year i multiplied by each action's energy value:

$$V_i(\bar{x}) = \textstyle\sum_{j=1}^{N} \sum_{k=1}^{i} R_{i,j,k}(\bar{x}) \cdot v_j, \tag{2}$$

- Finally, equation 3 represents the total yearly cost C_i, i.e., the sum of the costs on every resource for all actions executed on the $i-th$ year of the plan:

$$C_i(\bar{x}) = \textstyle\sum_{j=1}^{N} \sum_{l=1}^{L} x_{j,i} \cdot c_{j,l} \tag{3}$$

By definition, $V_i - C_i$ is $i-th$ year's total *cash flow*, and the NPV is the sum of all anual cash flows, adjusted by the internal return rate for every year. A solution with a bigger NPV means the EDCO will have a greater profit with that solution when compared to other solutions with lower NPV. So, the problem's objective is to maximize the NPV and the objective function is presented at equation 4:

$$max_{\bar{x}}(O(\bar{x})) = max_{\bar{x}} \left(\sum_{i=1}^{M} \frac{V_i(\bar{x}) - C_i(\bar{x})}{(1+r)^i} \right) \quad (4)$$

The problem's objective function must be maximized respecting some constraints. Those are:

- The yearly budget constraint (equation 5), which ensures that the solution's cost won't exceed the yearly budgets for each resource,
$$\sum_{j=1}^{N} x_{j,i} \cdot c_{j,l} \leq o_{i,l} \ \ \forall i, l, \quad (5)$$
- the market constraint (equation 6), which prevents the solution from surpassing any action's market,
$$\sum_{i=1}^{M} x_{j,i} \leq m_j \ \ \forall j, \quad (6)$$
- the yearly market constraint (equation 7), analogous to the market constraint, but now enforcing each action's yearly market,
$$x_{j,i} \leq u_{j,i} \ \ \forall j, i, \quad (7)$$
- the yearly reduction goal constraint (equation 8), which ensures that the yearly reduction goals won't be exceeded,
$$\sum_{j=1}^{N} \sum_{k=1}^{i} R_{i,j,k}(\bar{x}) \leq g_i \ \ \forall i, \quad (8)$$
- the dependency restriction (equation 9), which ensures that the solution will respect the dependency relations between actions for all actions of the plan,
$$\sum_{i=1}^{k} x_{d,i} \geq \sum_{i'=1}^{k} x_{j,i'} \cdot Q_{j,d} \quad \forall d \in D_j \quad \forall j, k. \quad (9)$$

The resulting problem obtained by the concatenation of equations 1 to 9, which models the EDCOs problem tackled in this paper, is presented in equation 10:

$$\begin{array}{lll}
\max\limits_{\bar{x}} & \sum_{i=1}^{M} \frac{V_i(\bar{x}) - C_i(\bar{x})}{(1+r)^i} & \textit{(NPV)} \\
\text{s. to} & \sum_{j=1}^{N} x_{j,i} \cdot c_{j,l} \leq o_{i,l} \ \ \forall i, l & \textit{(Yearly Budgets)} \\
 & \sum_{i=1}^{M} x_{j,i} \leq m_j \ \ \forall j & \textit{(Market)} \\
 & x_{j,i} \leq u_{j,i} \ \ \forall j, i & \textit{(Yearly Market)} \\
 & \sum_{j=1}^{N} \sum_{k=1}^{i} R_{i,j,k}(\bar{x}) \leq g_i \ \ \forall i & \textit{(Goals)} \\
 & \sum_{i=1}^{k} x_{d,i} \geq \sum_{i'=1}^{k} x_{j,i'} \cdot Q_{j,d} \ \ \forall d \in D_j \ \forall j, k & \textit{(Deps.)} \\
\text{where} & R_{i,j,k}(\bar{x}) = x_{j,k} \cdot e_{j,i-k+1} & \textit{(Energy Recovery)} \\
 & V_i(\bar{x}) = \sum_{j=1}^{N} \sum_{k=1}^{i} R_{i,j,k}(\bar{x}) \cdot v_j & \textit{(Yearly Profit)} \\
 & C_i(\bar{x}) = \sum_{j=1}^{N} \sum_{l=1}^{L} x_{j,i} \cdot c_{j,l} & \textit{(Yearly Cost)} \\
 & x_{j,i} \in N, \quad i \leq M, \quad j \leq N, \quad l \leq L &
\end{array} \quad (10)$$

B. The Knapsack Approach

The exact formulation of the problem has shown to be quite particular and no work addressing a similar problem was found in literature. In this work, we face it as a generalization of the well known knapsack problem, which has lots of practical applications. Even though our problem is not the exact generalization of the classical knapsack and its variants, we try to relate the hardness of those problems to our formulation of the EDCO's problem.

The *Knapsack Problem* (KP) is a well known problem on the literature consisting of the allocation of N items inside a knapsack with a maximum capacity c. Each item j has its weight w_j and profit p_j, and the objecive is to maximize the profit of the items in the knapsack, without exceeding its capacity. It can be formulated [2] as:

$$\begin{array}{lll}
\text{maximize} & \sum_{j=1}^{N} p_j \cdot x_j & \\
\text{subject to} & \sum_{j=1}^{N} w_j \cdot x_j \leq c, & \\
 & x_j \in \{0, 1\}, & j = 1, ..., N.
\end{array} \quad (11)$$

One possible generalization of the classical problem is to relax the restrictions in a way that more than one copy of the same item can be in the knapsack. That generalization is called *Bounded knapsack problem* (BKP) [2], and is usually solved with a transformation of the BKP instance to a KP instance with more variables [3], solving the resulting problem with KP techniques.

Another possible generalization can be achieved when we consider some kind of ordering between the items, or a dependency relation. That generalization is called the *partially-ordered knapsack problem* (POKP)[4]. In this problem, an item a may depend on item b, meaning that to be able to put item b in the knapsack, item a must also be on the knapsack. A review on knapsack problems with neighbour constraints, a generalization of the POKP, is presented in [5], and a real application is demonstrated in [6], where the Open Pit Block Sequencing problem is modelled using the POKP.

We may also consider a knapsack problem with several knapsacks available, each of them with its own capacity. This generalization is called *multiple knapsack problem* (MKP) [3]. In [7] the MKP is used to model the allocation of virtual mahines in a cloud computing environment, and the resulting model is solved using an Ant Colony Optimization based heuristic. In [8], the authors propose a Quantum-Inspired Evolutionary Algorithm to solve the MKP.

Finally, another generalization is obtained when we consider a knapsack with more than one dimension, like weight and volume. In this variation, each item has a diferent weigth $w_{j,r}$ on each dimension c_r. This generalization is known as the *mutidimensional knapsack problem* or *d-dimensional knapsack problem* (dKP) [3]. Due to its hardness and several pratical applications, the dKP is probably the most studied variation of the knapsack problem. For an overview on this variation the reader is directed to [9]. Recently, several heuristic approaches have been proposed to solve the dKP. For instance, [10] and [11] propose two Particle Swarm Optimization (PSO) algorithms to solve the dKP. The first proposes a binary PSO with accelerated particles and the second presents a binary PSO with acceleration varying with time. In [12] the Fruit Fly algorithm is adapted for binary variables, and the resulting implementation is tested using the literature's most used test instances (the OR-Library [13]). In [14], the authors investigate the efficiency of the application of linkage-learning on an Estimation of Distribution algorithm for the dKP.

All the problems mentioned above are hard to solve optimally, they are known as $\mathcal{NP}$-hard problems [3] and there are no known polynomial time algorithms to solve them, unless $\mathcal{P} = \mathcal{NP}$ [15]. For the classical knapsack problem there is a FPTAS (*Fully Polynomial Time Approximation Scheme*), while the other variants mentioned above are hard to aproximate and there are only PTAS (*Polynomial Time Approximation Schemes*) to solve them with a certain degree of approximation to the optimal solution, as shown in [4], [16], [17].

If we consider a knapsack with the characteristics of every gerenalization presented so far we can define a new generalization called *partially-ordered multidimensional multiple bounded knapsack problem* (POMMBKP), which is as far as we know, a novel generalization of the knapsack problem. Looking at the model described at section II-A it is possible to see that it is a generalization of the POMMBKP, when we make the following assumptions:

- the model's actions are the POMMBKP's items;
- the model's years are the POMMBKP's knapsacks;
- each resource an action consumes can be seen as a POMMBKP dimension;
- in the model, the actions can be executed more than one time, representing the bounded knapsack problem part of the POMMBKP;
- the loss reduction plan can be defined for many years, and each year may have several budgets. That may be seen as the multiple knapsacks problem and multidimensional knapsack problem respectively;
- actions may depend on other actions being executed previously, like on the partially-ordered knapsack problem.

Even with all the similarities, some differences between the model and the POMMBKP are noted. Firstly, the POMMBKP doesn't consider anything similar to the model's internal return rate. Also, only single dependencies are considered on the POMMBKP, while on the EDCO's problem multiple dependencies may occur, i.e. an action may require another action to be executed more than one time before. Finally, if we consider the model's goal as a resource, an action (item) can consume resources from more than one year (knapsack), something that doesn't happen on the POMMBKP.

Considering the EDCO's problem with an internal return rate equal to zero, only single dependencies and actions recovering energy only on the year they were executed, characteristics that can be representend on the model presented in equation 10, it can be seen as a generalization of the POMMBKP (for a proof the user is directed to [18]). From now on, we refer to the EDCO's problem as the *partially-ordered multidimensional multiple bounded knapsack with multiple dependencies, Spread Weights and Adjustment Rate* (POMMBKPDSA).

In [18] it is shown that the POMMBKPDSA is a generalization of the POMMBKP, so we can tell that the first is at least as difficult as the second. Since the POMMBK is a generalization of all the simple variations discussed before (BKP, POKP, MKP and dKP), it is also possible to say that the POMMBK is at least as difficult as any of them. We can conclude then that the POMMBKPDSA is also at least as difficult as any of those variations, so there is not any algorithm able to solve it in polynomial time, considering $\mathcal{P} \neq \mathcal{NP}$.

III. Computational Approach

To solve the POMMBKPDSA we used three approaches:

- An exact approach, using integer programming techniques;
- a heuristic based on a greedy strategy, using the truncated solution of the linear relaxation of the problem, called GALP;
- a Tabu Search based heuristic, also using the truncated solution of the linear relaxation of the problem, called TSLP.

The first approach used, the exact approach, was to solve the formulation of the problem described in equation 10 using the *branch-and-cut* method [19] implemented on the CPLEX solver.

Before explaining the other two approaches, it is important to explain the reason why both heuristics used the truncated solution of the LP-relaxed version of the problem as starting points. During our tests, we tried to use other alternatives as starting points for the heuristics, like empty solutions or randomly generated solutions. Still those alternatives never yielded better results than starting the heuristics from the truncated solution of the LP-relaxed version of the problem. In fact, we found out that this solution was on the worst case 97.7% of the solution of the LP-relaxed version of the problem, which is an upper bound, and at 99.65% on the average case. Being such a good starting point, and so easily obtainable by modern solvers, it seems a good choice to start the heuristic from this solution.

The second applied technique, the GALP, is a local search heuristic using a greedy strategy. Generally, greedy strategies have the disadvantage of not being able to escape any local optimum they reach. On the other hand, they usually converge faster than other methods, making them suitable alternatives when fast viable solutions are desired. Also, they are usually easy to understand and implement.

Fig 1 describes the greedy algorithm developed for the POMMBKPDSA. The heuristic receives as initial solution the truncated solution of the LP-relaxed version of the problem. It then tries to improve it by iteratively adding items to the knapsacks until no more additions are possible. It is divided in three phases: On the first phase, the algorithm generates a list of all possible allocations which may be made on a solution. Those alocations are all the possible combinations of item vs. knapsack, and represent the addition of a certain item to a knapsack. That list is then sorted by decreasing order of each alocation's efficiency, i.e. how much each alocation contributes to the increase of the objective function. In the last phase, that sorted list of alocations is used to iterativelly improve the initial solution.

At line 2, GenerateAlocations() function generates a list with all the possible alocations of items in knapsacks. For

Input: Initial Solution: $S_{initial}$
Output: Best Solution: S_{best}
1: $S_{best} \leftarrow S_{initial}$
2: $AlocationList \leftarrow GenerateAlocations()$
3: $AlocationList \leftarrow DecreasingOrder(AlocationList)$
4: $Improved \leftarrow$ **true**
5: **while** $Improved$ **do**
6: $Improved \leftarrow$ **false**
7: **for** $Alocation \in AlocationList$ && $not\ Improved$ **do**
8: **if** $isViable(S_{best}, Alocation)$ **then**
9: $S_{best} \leftarrow S_{best} + Alocation$
10: $Improved \leftarrow$ **true**
11: **end if**
12: **end for**
13: **end while**
14: **return** S_{best}

Fig. 1: Greedy Algorithm

Input: Initial Solution: $S_{initial}$
Output: Best Solution: S_{best}
1: $S_{current} \leftarrow S_{initial}$
2: $S_{best} \leftarrow S_{initial}$
3: $ListaTabu \leftarrow \emptyset$
4: **while** $\neg stoppingCondition$ **do**
5: $moves \leftarrow Neighborhood(S_{current})$
6: $move \leftarrow BestNonTabuMove(moves)$
7: $S_{current} \leftarrow S_{current} + move$
8: $TabuList \leftarrow TabuList + move$
9: **if** $S_{current} \geq S_{best}$ **then**
10: $S_{best} \leftarrow S_{current}$
11: $ResetTabuList()$
12: **end if**
13: **end while**
14: **return** S_{best}

Fig. 2: Tabu Search

example, for an instance of the problem with two knapsacks and three items, the list generated would be {(1,1), (1,2), (2,1), (2,2), (3,1), (3,2)}, where each pair from the list represents adding one copy of an item to a knapsack. On the sequence, on line 3 that list is ordered by descending order of efficiency, i.e. how much each item contributes to the increase on the objective function's value. The initial solution is then iteratively improved on the loop between lines 5 and 13. At each step, the alocation list is searched for the alocation with the highest efficiency which leads to a viable solution. Once that alocation is found, the currently best solution is updated with the alocation (line 9) and the loop continues, until no alocation can lead to a viable solution. At that point, the algorithm stops and returns the best solution found.

The last applied heuristic, the TSLP, is an implementation of the metaheuristic Tabu Search [20]. This metaheuristic has been heavilly used on the last decades to tackle Combinatorial Optimization problems, has already been applied on the dKP in [21] and for some time the Tabu Search presented at [22] and improved at [23] was the method which obtained the best results for the dKP intances commonly used in the litarature.

Like the greedy algorithm previously presented, the Tabu Search is also essentially a local search method, so it should also present a tendency to become stuck at local optima. To work around this problem, the Tabu Search allows that during the search, when no solution better than the current one is found, a worse solution may be chosen with the spectation that this choice will lead to better solutions on the long run. Also, in this method, the characteristics of all the choices made during the search may be taken in consideration when making the next choice. Because of those characteristics, the Tabu Search is usually able to escape local optima and find better solutions than other simple local searches.

Before explaining the implemented heuristic, it's useful to explain some terms which will be used ahead. The first of them are the *moves*. Moves are the operations applied on a solution which takes them to another solution. The set of solutions which can be reached from solution x by applying one of the possible movements is called x's *Neighborhood*. One characteristic component of the Tabu Search is the *Tabu List*. This list usually contains moves or characteristics that lead to unwanted solutions, and is used during the search to forbid certain moves as a way to escape local optima and hopefully guide the search towards better solutions.

Fig. 2 describes the implemented heuristic. It receives the initial solution and tries to improve that solution by iteractively searching its neighborhood.

The most relevant part of the heuristic happens on the loop between lines 4 and 13. Firstly, all the moves leading to neighbours of the current solutions are obtained at line 5. At line 6 the function *BestMoveNotTabu()* searches the movement list for the move which will lead to the greatest value for the objective function and is not forbidden by the Tabu List. The current solution is then updated with the chosen move (line 7), the Tabu List is also updated with the chosen move (line 8) and finally if the current solution is better than the best solution found so far, the best solution is updated with the current solution (line 10) and the tabu list is reset.

An important aspect to be considered when implementing a Tabu Search is how to implement and manage the Tabu List (TL). In this work, a dynamic TL management method was implemented based on the Reverse Elimination Method (REM) [24]. This method leads to an exact tabu status, meaning that any move present in the TL leads to a solution that was already visited by the search. The method works by adding the move made at each iteraction to the Tabu List. When a new move is added to the Tabu List, the list is traced in reverse order to build an Active Tabu List (ATL), which contains every move that will lead to a solution already visited during the search. Now, in order to check if a move is forbidden, all we have to do is check if it is contained in the ATL. When a new global optimum is found during the search, the TL is cleared. For more details on the Reverse Elimination Method the reader is directed to [24].

IV. Experimental Data

In order to test the hardness of the POMMBKPDSA and assess the quality of the solutions obtained by the methods proposed on the previous section, a large number of instances of the problem are needed to make useful conclusions. Unfortunatelly, our local EDCO wasn't able to provide that number of instances, so we had to develop an instance generator to create a sufficient amount of artificial instances to run the tests. The generator creates instances according to the formulation of the POMMBKPDSA, with parameters that can be adjusted to build instances of different sizes and characteristics. It receives as parameters:

- the number of years (knapsacks);
- the number of actions (items);
- the number of budgets or resources (dimensions);
- the level of correlation between each action's cost and profit (α).

After those input parameters are set, the instance generation procedure follows three steps. At first the actions characteristics are generated (recovery curve, costs, markets and electricity value) using uniform distributions with variations similar to those found in real world applications. The second step is to compute yearly goals and yearly budgets, based on total sums of the characteristics of the actions. The last step generates some random dependencies between actions. This is done avoiding cyclical dependencies, as any action involved in a cyclical dependency would never be chosen.

V. Experimental Results

In order to evaluate the hardness of the instances created with the generator and the effectiveness of the proposed solution approaches, computational tests were conducted using both heuristics and the generic solver CPLEX.

Firstly, the created instances were solved with the CPLEX solver version 12.5.0, in order to verify their hardness. The solver was used in its default configuration. Then, the GALP and TSLP algorithms were executed on the same instances. While GALP doesn't use any adjustable parameter, the TSLP has the stopping condition, which was 10000 iterations in our tests. All the tests were executed on Intel Core i5-3570 @ 3.40 GHz machines with 8GB RAM memory and executing Ubuntu 13.04. Both heuristics were implemented in Java and run on the 1.7 JVM.

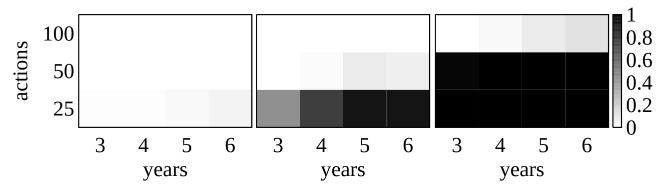
To execute the tests a suite of artificial benchmark instances was created using the generator presented at the last session. The instances were created for all combinations of 3 to 6 years, 25, 50 and 100 actions, 1, 2 and 4 resources and correlation levels (α) of 0.0, 0.1 and 1.0. Those numbers of years, actions and resources were chosen to simulate the dimensions of the instances expected to be found in practice. The α values were chosen to verify if this generalization of the KP is also sensitive to the correlation level on the items profits and costs, as predicted by the KP literature [25], and represent respectivelly strongly, weakly and uncorrelated instances. Therefore, 100 instances for each possible parameter combination were created, totaling 10800 instances.

Next session shows a hardness analysis using the results of the CPLEX tests. Then, the results of the tests with the two implemented heuristics are discussed.

A. Problem Hardness

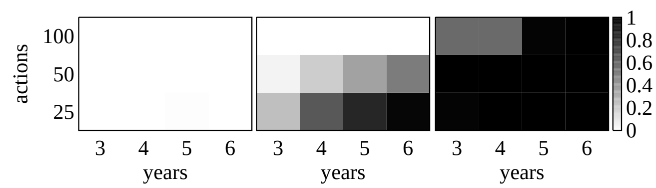
The first battery of tests involved solving the generated instances with the CPLEX solver, to check if they were indeed hard to solve. During preliminary tests, while the CPEX solver was able to fastly solve some instances, it took a lot of time to solve some others. So to be able to continue the experiments, an alternative strategy was used: instances that took more than 20 minutes to be solved were considered prohibitive for the continuity of the experiment, so when an instance reached this threshold, it was interrupted and the current solution saved.

Figures 3, 4 and 5 show the results of the first tests, considering the interruption rate. Each of the three figures represents a level of correlation, while each heatmap shows the instances with certain number of resources. The colors on the heatmaps represent the ratio of interrupted instances during the CPLEX execution, varying from the combinations of actions/years where no instance was interrupted (paler squares) to the combinations where all instances were interrupted (darker squares).



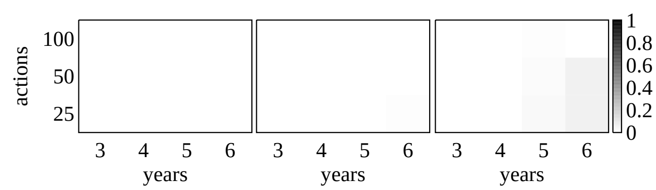
(a) 1 resource (b) 2 resources (c) 4 resources

Fig. 3: Ratio of strongly correlated instances interrupted ($\alpha = 0.0$).



(a) 1 resource (b) 2 resources (c) 4 resources

Fig. 4: Ratio of weakly correlated instances interrupted ($\alpha = 0.1$).



(a) 1 resource (b) 2 resources (c) 4 resources

Fig. 5: Ratio of uncorrelated instances interrupted ($\alpha = 1.0$).

Comparing figures 3 and 4 to figure 5 one can see the first two figures are darker than the last, meaning that the correlation level indeed affects the instances hardness. As expected, uncorrelated instances are easier for the CPLEX to solve, with just a few instances with bigger dimensions being interrupted. However, comparing only figure 3 to figure 4, it appears that weakly correlated instances are harder to solve than strongly correlated instances, as opposed to the expected.

Concerning quantity of years and resources of the instances, the obtained results confirm what was expected, since figures 3 to 5 show that the bigger the instances are on this two parameters, the harder it gets for the CPLEX solver to solve them to optimality. Still, comparing the three figures observing the hardness in relation to the variation on the number of actions, it seems that instances with fewer actions are harder to solve to optimallity.

In relation to the time taken to solve the instances, the CPLEX took on average 380 seconds to solve the instances. The average gap was 0.01%, meaning that the CPLEX found solutions that were, in average, at least 99.99% of the optimal ones.

B. Solutions Quality

To analyse the quality of the solutions obtained by the heuristics, the same instances previously solved with CPLEX were now solved with the GALP and TSLP, and the solutions found were compared to the ones obtained by CPLEX before reaching the time limit. Figures 6 to 8 show the results for the TSLP. The heatmaps follow the same configuration of the CPLEX tests, but now showing the average quality of the solutions. The color scale represents the average ratio between the solutions found by the TSLP and the best known solution, in other words, darker tones indicate solutions with better quality were found by the TSLP. The same representations are used on figures 9 to 11 to present the results obtained with GALP.

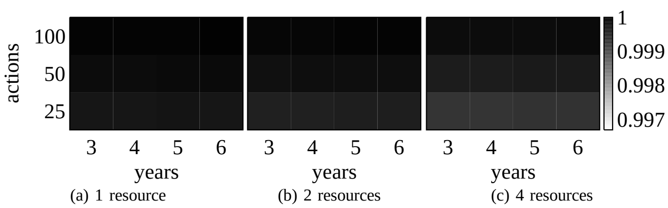


Fig. 6: TSLP solution quality on strongly correlated instances ($\alpha = 0.0$).

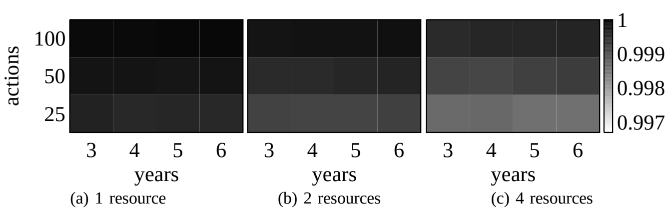


Fig. 7: TSLP solution quality on weakly correlated instances ($\alpha = 0.1$).

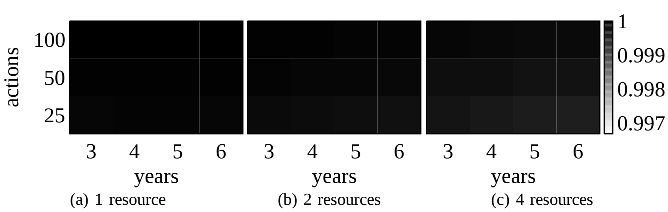


Fig. 8: TSLP solution quality on uncorrelated instances ($\alpha = 1.0$).

Looking at figures 6, 7 and 8, which show the results of the TSLP tests, some impact of the correlation level on the quality of the obtained solutions can again be seen, as the heatmaps for the instances with some level of correlation are paler than the ones for the uncorrelated instances. Specially on the instances with weak correlation (figure 7), the algorithm seems to obtain the worst solutions.

Comparing the results in respect to the number of resources, it can be seen that instances with more resources are harder to solve. That influence can be seen mainly on figure 7. Once again, an observation of the quality of the solutions related to the amount of actions on the instance shows that the TSLP also obtained worse solutions on instances with fewer actions. The influence of the number of years in those tests was too weak to be considered significant.

Figures 9, 10 and 11 present the results of the GALP heuristic. Once again, it is possible to see a reduction on the quality of the solutions found when the quantity of resources or level of correlation on the instances are increased. Besides, increasing the number of actions on the instances also enabled GALP to find better solutions, and the influence of the number of years was also not significant, as observed on the TSLP tests.

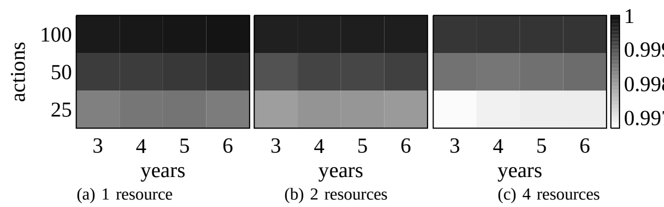


Fig. 9: GALP solution quality on strongly correlated instances ($\alpha = 0.0$).

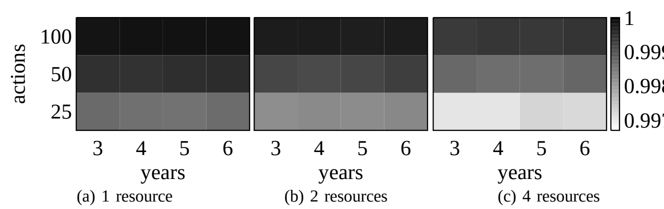


Fig. 10: GALP solution quality on weakly correlated instances ($\alpha = 0.1$).

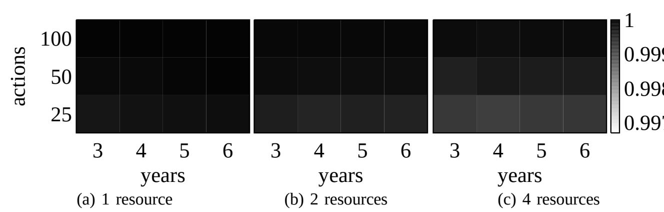


Fig. 11: GALP solution quality on uncorrelated instances ($\alpha = 1.0$).

Observing the figures a comparison between the two heuristics can also be made. By comparing figures 6, 7 and 8 to figures 9, 10 and 11, the first group of figures show a darker tone than the second. It indicates that on average the TSLP was able to find better solutions than the GALP on the tested instances. All the experiments results are available under request to the authors.

Even considering that the heuristics starting point was already a good solution, since it was on average 99.652% of the best known solution, the heuristics were able to improve the solutions. On average, the solutions found by the GALP were 99.912% of the best known solutions, and the TSLP obtained even better results, with it's solutions being 99.967% of the best known ones, on average.

The TSLP had a maximum execution time of 2 minutes, with an average of 30 seconds. GALP had a negligible running time, below 1 second for all instances.

VI. Conclusion

The main objectives on this work were (1) to understand and formally model the EDCO's loss reduction problem, (2) test the hardness of the created model and (3) solve the model with exact and heuristic approaches.

The electricity loss reduction problem was modelled as a generalization of the famous knapsack problem. So the first goal was met with the creation of a mathematical model combining characteristics of the Bounded knapsack problem, the multidimensional knapsack Problem, the multiple knapsack problem and the partially-odered knapsack problem. While reaching this first goal, two generalizations of the knapsack problem were created, the POMMBKP and the POMMBKPDSA, the first being a generalization of all those knapsack problem variations and the second being a generalization of the first. No work concerning any of these two new knapsack problem versions was found on the literature, and we believe that, due to their generality, they may be used to model several real world problems with greater details.

Due to the lack of real world instances for the EDCO's problem (the POMMBKPDSA), another contribution was made in the form of an artificial instance generator and a set of benchmark instances for the problem, available under request to the authors. As a way to achieve objective (2) those bechmark instances were solved with the CPLEX solver. These experiments demonstrated that with the increase of some dimensions of the instances, the exact solver is no longer able to prove the optimal solution within a limited time. It was shown that this incarnation of the knapsack problem, even if not exactly as predicted by literature, is also sensitive to the instances correlation level.

To achieve objective (3), besides the CPLEX solver two heuristics were proposed, the first based on a greedy approach (GALP) and the second based on the metaheuristic Tabu Search (TSLP). The heuristics were used to solve the benchmark instances and their solutions were compared to the best known ones. Both heuristics where able to find good quality solution, quite close to the best known. Particularly, the TSLP obtained better results concerning the quality of the solutions found, but at a greater time cost.

However it is worthy to note that the best solutions were still found by the CPLEX solver. Besides, even when the CPLEX's solving process was interrupted before the end, on the worst cases the solution was at most at 0.35% of the optimal solution. Being the method which found the best solutions, even in the cases it was interrupted, the CPLEX is probably the best option to solve instances of the problem with the dimensions expected to be found in practice.

Acknowledgement

The authors wish to thank EDP Energias do Brasil S.A. for cooperation in this research.

References

[1] G. A. S. Caputo and E. d. Alencastro, "Technical note n.0251/2012-sre/aneel." http://www.aneel.gov.br/cedoc/nreh20121326.pdf, 2012.

[2] D. Pisinger, *Algorithms for knapsack problems*. PhD thesis, University of Copenhagen, 1995.

[3] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*. Springer, 2004.

[4] S. G. Kolliopoulos and G. Steiner, "Partially-ordered knapsack and applications to scheduling," in *Algorithms – ESA 2002* (R. Möhring and R. Raman, eds.), pp. 612–624, Springer Berlin Heidelberg, 2002.

[5] G. Borradaile, B. Heeringa, and G. Wilfong, "The knapsack problem with neighbour constraints," *Journal of Discrete Algorithms*, vol. 16, no. 0, pp. 224 – 235, 2012.

[6] W. Lambert and A. Newman, "Tailored lagrangian relaxation for the open pit block sequencing problem," *Annals of Operations Research*, vol. 222, no. 1, pp. 419–438, 2014.

[7] S. R. M. Amarante, F. M. Roberto, A. R. Cardoso, and J. Celestino, "Using the multiple knapsack problem to model the problem of virtual machine allocation in cloud computing," *2013 IEEE 16th International Conference on Computational Science and Engineering*, vol. 0, pp. 476–483, 2013.

[8] C. Patvardhan, S. Bansal, and A. Srivastav, "Balanced quantum-inspired evolutionary algorithm for multiple knapsack problem," *International Journal of Intelligent Systems and Applications*, vol. 6, no. 11, pp. 1–12, 2014.

[9] A. Freville, "The multidimensional 01 knapsack problem: An overview," *European Journal of Operational Research*, vol. 155, no. 1, pp. 1 – 21, 2004.

[10] Z. Beheshti, S. Shamsuddin, and S. Yuhaniz, "Binary accelerated particle swarm algorithm (bapsa) for discrete optimization problems," *Journal of Global Optimization*, vol. 57, no. 2, pp. 549–573, 2013.

[11] M. Chih, C. J. Lin, M. S. Chern, and T. Y. Ou, "Particle swarm optimization with time-varying acceleration coefficients for the multidimensional knapsack problem," *Applied Mathematical Modelling*, vol. 38, no. 4, pp. 1338 – 1350, 2014.

[12] L. Wang, X. L. Zheng, and S. Y. Wang, "A novel binary fruit fly optimization algorithm for solving the multidimensional knapsack problem," *Knowledge-Based Systems*, vol. 48, no. 0, pp. 17 – 23, 2013.

[13] P. Chu and J. Beasley, "A genetic algorithm for the multidimensional knapsack problem," *Journal of Heuristics*, vol. 4, no. 1, pp. 63–86, 1998.

[14] J. P. Martins, C. M. Fonseca, and A. C. B. Delbem, "On the performance of linkage-tree genetic algorithms for the multidimensional knapsack problem," *Neurocomputing*, vol. 146, no. 0, pp. 17 – 29, 2014.

[15] M. R. Garey and D. S. Johnson, "Strong np-completeness results: Motivation, examples, and implications," *J. ACM*, vol. 25, pp. 499–508, July 1978.

[16] J. Puchinger, G. R. Raidl, and U. Pferschy, "The core concept for the multidimensional knapsack problem," in *Evolutionary Computation in Combinatorial Optimization* (J. Gottlieb and G. R. Raidl, eds.), pp. 195–208, Springer Berlin Heidelberg, 2006.

[17] M. Dawande, J. Kalagnanam, P. Keskinocak, F. S. Salman, and R. Ravi, "Approximation algorithms for the multiple knapsack problem with assignment restrictions," *Journal of Combinatorial Optimization*, vol. 4, no. 2, pp. 171–186, 2000.

[18] J. C. H. Moreira, "Uma abordagem com multi-mochilas multidimensionais para o problema de alocação de ações de redução de perdas na distribuição de energia," Master's thesis, Universidade Federal do Espírito Santo, 2015.

[19] M. Padberg and G. Rinaldi, "A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems," *SIAM review*, vol. 33, no. 1, pp. 60–100, 1991.

[20] F. Glover, "Tabu search - part 1," *ORSA Journal on Computing*, vol. 1, no. 3, pp. 190–206, 1989.

[21] F. Dammeyer and S. Vo, “Dynamic tabu list management using the reverse elimination method,” *Annals of Operations Research*, vol. 41, no. 2, pp. 29–46, 1993.
[22] M. Vasquez and J. Hao, “A hybrid approach for the 01 multidimensional knapsack problem,” in *International Joint Conference on Artificial Intelligence*, pp. 328–333, 2001.
[23] M. Vasquez and Y. Vimont, “Improved results on the 0-1 multidimensional knapsack problem,” *European Journal of Operational Research*, vol. 165, pp. 70–81, 2005.
[24] F. Glover, “Tabu search - part 2,” *ORSA Journal on Computing*, vol. 2, no. 1, pp. 4–32, 1990.
[25] D. Pisinger, “Where are the hard knapsack problems,” *Computers & Operations Research*, vol. 32, no. 9, pp. 2271–2284, 2005.