A Frank-Wolfe Based Algorithm for Robust Discrete Optimization Under Uncertainty

Chifaa Al Dahik*†‡, Zeina Al Masry*†, Stéphane Chrétien§, Jean-Marc Nicod *†, Landy Rabehasaina*‡
*Université Bourgogne Franche-Comté/CNRS/ ENSMM, Besançon, France
† FEMTO-ST institute, Besançon, France
‡ Laboratoire de Mathématiques de Besançon, Besançon, France
§ Laboratoire ERIC, UFR ASSP, Université Lyon2, Lyon, France

***Abstract*—This paper addresses a class of robust optimization problems whose inputs are correlated and belong to an ellipsoidal uncertainty set, which is known to be NP-Hard. For that, we propose an efficient heuristic scalable approach based on the iterative Frank-Wolfe (FW) algorithm. In our approach, we take a radically different perspective on FW by looking at the exploration power of the integer inner iterates of the method. Our main discovery is that, for small dimensional instances, our method is able to provide the same optimal integer solution as an exact method provided by CPLEX, after no more than a few hundred iterations. Moreover, as opposed to the exact method, our FW-guided integer exploration approach applies to large scale problems as well. Our findings are illustrated by comprehensive numerical experiments. We focus on two target applications, the robust shortest path problem as a first test case, and the robust clustering as a real application in a PHM context and data analysis.**

***Index Terms*—Uncertainty, Robust Clustering, Robust Discrete Optimization, Ellipsoidal uncertainty set**

I. Introduction

Prognostics and Health managements (PHM) is a discipline that allows to understand the behavior of a given system and to provide an adapted decision in order to manage the state of health of this system. PHM is based on several pillars such as data analysis, diagnostic, prognostics and decision making [1], [2]. Clustering techniques are one of the different tools used in PHM: It can be used, for example, to classify the remaining useful life (RUL) of several machines for scheduling purposes [3]–[5]. Clustering is a common optimization problem that consists in finding the best groups that are close to some cluster centers. It has many variants such as the k-median, the k-means, or new clustering definitions that do not require the number of cluster centers [6]. When the distances between the objects are known, there exist algorithms that propose optimal clustering. However, in the case where these distances are uncertain, a robust solution for that cluster is needed. This solution is not supposed to be optimal in all cases, but it accommodates with potential fluctuations of the input data set. Up to our knowledge, a robust version of the clustering problem does not exist so far, despite its relevance and impact on PHM, data analysis and other disciplines.

The problem of finding robust solutions in view of decision-making has received increased attention lately, due to its practical relevance. While some studies prove theoretical solvability for different definitions of robustness, others suggest exact and heuristic methods for computing a robust solution [7]. In real life problems, i.e., large size problem, finding a solution for this problem using branch-and-bound techniques can be difficult and sometimes even impossible. Then, it is mandatory to develop some scalable methods for robust combinatorial optimization.

One way of addressing the problem is to relax the underlying constraint set by considering its convex hull. However, solving the problem on the convex hull might not be relevant for decision-making, especially when the decision needs a feasible solution. Nevertheless, some methods, such as the Frank-Wolfe algorithm, implement intermediate steps that solve at each iteration a version where the objective function is linearized. In the case where these intermediate steps provide feasible solutions (such as for integer linear programs (ILP) with total unimodularity), the Frank-Wolfe algorithm exhibits feasible solutions along the trajectory.

This paper aims to study how these intermediate feasible solutions can help solving the original combinatorial robust optimization problem. For the ease of the understanding of our approach, the chosen motivating example is the robust shortest path problem. Our main empirical findings show in the case of the robust shortest path problem that our Frank-Wolfe-based algorithm swiftly discovers the optimal robust solution along its process, despite the intrinsic worst case difficulty of the considered robust counterpart. Comparisons with CPLEX [8], an optimization software package based on branch-and-bound methods, illustrate our discovery. These unexpected results open the way to further investigations into the problem of certifying the exactness of the exhibited solutions, via e.g., duality tools[1]. Then, an extension of the proposed algorithm is presented for the robust clustering problem.

This paper is organized as follows: Section II presents the state-of-the-art of the robust discrete optimization. Section III gives the problem statement addressed by the paper. Section IV details an exact method of the problem solving. Section V exhibits the suggested scalable heuristic approach and Section VI presents a numerical example, namely the

[1]see forthcoming extended version of this work to be posted on ArXiv.

robust shortest path problem. Section VII proposes a robust version of the clustering problem. Section VIII ends the paper by presenting a conclusion and future work.

II. Related work

In the literature, many definitions of robustness were proposed, such as absolute robust solution [9], robust deviation [9], relative robust solution [9], robust two-stage problems [10], K-adaptability [11], light robustness [12], etc. We here are interested in the absolute robustness criterion since it is relatively easy to evaluate from a computational perspective. Recall that the absolute robust solution is feasible for every scenario in the set of all the relevant realizations of the uncertainty parameters, and it is worst-case optimal. In addition, a choice of the uncertainty set should be made such as discrete, polyhedral (interval) and ellipsoidal sets [7]. The ellipsoidal uncertainty set is less conservative than interval uncertainty set [13], since it eliminates the extreme points of the set that are not likely to occur. It also permits to control how much one is willing to accept the risk (see Section III). Moreover, it allows to model correlations between the coefficients of the cost. Note that in reality significant correlations between the cost vector components may occur. This makes the problem computationally challenging. Approaches in the non-correlated case are proposed by Baumann *et al.* in [14] and by Bertsimas *et al.* in [15]. In order to fit with real applications, we here consider the ellipsoidal uncertainty set in the correlated case.

The optimization problem considered in this paper is NP-hard, i.e., it is unlikely that there can be developed a polynomial-time algorithm for finding an optimal solution, and to the best of our knowledge, there are not many numerical methods for solving this type of problems. In [16], a Lagrangean decomposition approach for the branch-and-bound algorithm has been suggested, but this approach is not scalable. On the other hand, a Frank–Wolfe based branch-and-bound algorithm has been devised for a similar problem in [17], where it is applied to a general ellipsoidal uncertainty set. The problem, written in the form of a mixed integer second order cone programming problem (MISOCP), can be solved numerically by a MISOCP solver like CPLEX. This type of solvers is usually used as a reference for comparing obtained solutions and computation efficiency of the above approaches.

The main result of this paper is the proposition of a heuristic algorithm inspired from Frank-Wolfe's algorithm for solving robust discrete optimization problems as presented in the sequel. It is worth to mention that, up to our knowledge, there does not exist any adaptation of Frank-Wolfe's algorithm for discrete problems.

III. Problem statement

We consider general combinatorial optimization problems of the form:

$$\min_{x \in X} c^T x \tag{1}$$

where $X \subseteq \mathbb{N}^m$ is the set of the feasible solutions, and where the cost vector $c \in \mathbb{R}^m$ is subject to uncertainty, i.e., it has more than one possible realization. Let U be an uncertainty set included in $\mathbb{R}^m$ ($U \subseteq \mathbb{R}^m$). The absolute robust counterpart of Problem (1) is then defined as:

$$\min_{x \in X} \max_{c \in U} c^T x. \tag{2}$$

Problem (2) is motivated as follows. If we suppose that c takes its values in U, then solving (2) amounts to finding the optimal solution in X corresponding to the worst possible cost. If we consider that the cost coefficient vector c has a multinormal distribution with expectation $\mu \in \mathbb{R}^m$ and covariance matrix $\Sigma \in \mathbb{R}^{m \times m}$, which is a reasonable assumption, an interesting uncertainty set to consider is the following ellipsoid:

$$U = \{c \in \mathbb{R}^m; (c-\mu)^T \Sigma^{-1} (c-\mu) \leq \Omega^2\}. \tag{3}$$

Indeed, U is a confidence set for c, i.e., c belongs to U with probability $1-\alpha \in [0,1]$, where $\Omega > 0$ is written as $\Omega_\alpha = \chi^2_m(1-\alpha)$ and $\chi^2_m(1-\alpha)$ refers to the quantile function for probability $1-\alpha$ of the chi-squared distribution with m degrees of freedom . In that case, the parameter Ω describes the level of confidence in solving the corresponding problem (2), i.e., the risk the user is willing to take. If $\Omega = \Omega_\alpha$ is small, i.e., α is close to 1, the user is willing to accept the risk to obtain a better solution. While a user who chooses a bigger value for Ω prefers to be secured in more cases, even if these cases are less likely to occur.

If U is defined by the set (3), problem (2) can be rewritten as a non-linear optimization problem:

$$\min_{x \in X} \mu^T x + \Omega \sqrt{x^T \Sigma x}. \tag{4}$$

This reformulation can be done using conic duality or by means of a substitution that reduces the task to a linear maximization problem over a ball (for more details see Section 2.2.1.1 of [16]). We see that in the special case of the ellipsoidal uncertainty set, finding a robust solution is reduced to solving a deterministic optimization problem. In addition, one could interpret (4) as a weighted sum of the mean $\mu^T x$ and the risk $\sqrt{x^T \Sigma x}$, which is called a mean-risk optimization problem [18].

IV. Method for computing an optimal solution

There exists another formulation of problem (4) that permits us to solve it using existing algorithms which can be found in [16]. First note that we are able to write the objective function as $\mu^T x + \sqrt{x^T \Sigma x}$, by replacing Σ by $\Omega^2 \Sigma$, in order to simplify its expression. Problem (4) can be written as a mixed integer second order cone programming problem (MISOCP). Since Σ is positive symmetric semi-definite, then we can write $\Sigma = \Sigma^{\frac{1}{2}} \Sigma^{\frac{1}{2}}$, with $\Sigma^{\frac{1}{2}}$ symmetric, and we obtain:

$$\begin{aligned} & \min \mu^T x + \sqrt{x^T \Sigma x} \\ & \text{s.t. } x \in X \\ \iff \quad & \min \mu^T x + \sqrt{x^T \Sigma^{\frac{1}{2}} (\Sigma^{\frac{1}{2}})^T x} \\ & \text{s.t. } x \in X. \end{aligned} \tag{5}$$

With a change of variable and after adding a variable that transforms the objective function to a linear one, (5) becomes:

$$\begin{aligned} & \min \mu^T x + z \\ & \text{s.t. } \|y\|_2 \leq z \\ & y = (\Sigma^{\frac{1}{2}})^T x \\ & x \in X, y \in \mathbb{R}^m, z \in \mathbb{R}_+. \end{aligned}$$

Finally, noticing that a cone constraint is revealed, we conclude that (5) is equivalent to:

$$\begin{aligned} & \min \mu^T x + z \\ & \text{s.t. } (y, z)^T \in K_{m+1} \\ & y = (\Sigma^{\frac{1}{2}})^T x \\ & x \in X, y \in \mathbb{R}^m, z \in \mathbb{R}_+, \end{aligned} \tag{6}$$

with $K_{m+1} = \{x \in \mathbb{R}^{m+1}; \|(x_1, \ldots, x_m)^T\|_2 \leq x_{m+1}\}$ being a *second order cone*. The same problem without the integrality condition corresponds to a second order cone program (SOCP) and can be solved in a polynomial time [19]. Thus, we easily obtain a lower bound for a branch-and-bound method to get an optimal solution. There exists a MISOCP solver in CPLEX [8] that gives an optimal solution of the addressed problem using the formulation (6). But we mention that for problems of large size, the processing time of branch-and-bound methods may become considerable, since the size of the tree may grow exponentially. Thus proposing a heuristic algorithm seems mandatory. Such algorithm is presented in the following section.

V. Scalable suggested heuristic algorithm

The proposed heuristic algorithm is a variant of the Frank-Wolfe algorithm [20]. It requires some specific assumptions which are listed first. Then we recall the classical Frank-Wolfe algorithm. Finally, we describe the algorithm in detail.

A. Assumptions

From now on, we consider the robust counterparts (4) of problems in the form (1) that verify the following assumptions:

(A1) For any real-valued vector a (not necessarily with positive entries), there exists an efficient algorithm to solve $\min_{x \in X} a^T x$;

(A2) For any real-valued vector a, there exists a solution for $\min_{x \in \text{Conv}(X)} a^T x$ that belongs to X, where $\text{Conv}(X) \subset \mathbb{R}^m$ is the convex hull of X;

(A3) The vector with zeros in all entries $0_{\mathbb{R}^m}$ does not belong to X.

An example for which these assumptions are verified is the robust shortest path problem: there exist polynomial time algorithms to optimally solve the deterministic shortest path problem with real valued costs, hence **(A1)** is satisfied. **(A2)** is verified since the incidence matrix of the graph in that problem is totally unimodular. The vector $0_{\mathbb{R}^m}$ is never a path, hence **(A3)** is satisfied. Another example is the workforce planning problem(see [21, Section 0.8]). For more details about totally unimodular matrices, see [21].

B. The classical Frank-Wolfe algorithm

Let f be a real valued, convex and continuously differentiable function defined on a compact convex D. We consider in this section general constrained convex optimization problems of the form

$$\min_{x \in D} f(x).$$

For such optimization problems, one of the simplest and earliest known iterative optimizers is given by the Frank-Wolfe method, also known as *conditional gradient method*, described in Algorithm 1.

Algorithm 1 Frank-Wolfe 1956 [20]

Let $x^{(0)} \in D$
for $k = 0$ to K **do**
  compute $s^{(k)} := \underset{s \in D}{\operatorname{argmin}} \nabla f(x^{(k)})^T s$
  update $x^{(k+1)} := (1 - \gamma^{(k)}) x^{(k)} + \gamma^{(k)} s^{(k)}$
end for

Note that the step size $\gamma^{(k)}$ in Algorithm 1 admits several variants. The simplest one is $\gamma^{(k)} = \frac{2}{k+2}$, a more advanced one is the step size by line search:

$$\gamma^{(k)} = \underset{\alpha \in [0,1]}{\operatorname{argmin}} f((1 - \alpha) x^{(k)} + \alpha s).$$

This algorithm proceeds as follows. In each step, at a current position $x^{(k)}$, the algorithm moves in the direction of a minimizer of the linearization of the objective function (see [22]). The main advantage of this algorithmis that there is no need for projection of the updated position on the convex set D at each step k. Another interesting fact on the computational side is that at each step, we only need to solve an optimization problem with a linear objective function.

C. A Frank-Wolfe based algorithm

Recall that our objective now is to solve:

$$\min_{x \in X} g(x) \tag{7}$$

where

$$g(x) = \mu^T x + \Omega \sqrt{x^T \Sigma x}.$$

Note that Problem (7) is a constrained integer non-linear problem. In order to define the gradient needed in Algorithm 2, it is reasonable to suppose that the covariance matrix Σ is symmetric positive definite (as in [16]) and not only symmetric positive semi-definite. This means that $x^T \Sigma x$ is null only if x

equals $0_{\mathbb{R}^m}$ that does not belong to X thanks to **(A3)**. So for all x in X, the gradient of g at x is computed as follows:

$$\nabla g(x) = \mu + \Omega \frac{\Sigma x}{\sqrt{x^T \Sigma x}}.$$

One should notice that we are not capable of using the Frank-Wolfe algorithm directly to solve (7), since the constraint set X is discrete and so the problem is not convex. Instead, we suggest Algorithm 2 that we denote as DFW referring to Discrete Franke-Wolfe.

Algorithm 2 DFW: a Frank-Wolfe based algorithm to solve (7)

1: $x^{(0)}$ a random feasible solution, $\varepsilon > 0$ close to zero, K a maximum number of iterations.
2: $k := 1$
3: stop := false
4: **while** $k \leq K$ and $\neg$stop **do**
5: **if** $g(x^{(k-1)}) - g(x^{(k)}) < \varepsilon$: **then**
6: stop := true
7: **else**
8: $s^{(k)} \in \underset{y \in \text{Conv}(X)}{\text{argmin}} \nabla g(x^{(k)})^T y$, with $s^{(k)} \in X$
9: $\gamma^{(k)} = \underset{\alpha \in [0,1]}{\text{argmin}}\, g(x^{(k)} + \alpha(s^{(k)} - x^{(k)}))$
10: $x^{(k+1)} = x^{(k)} + \gamma^{(k)}(s^{(k)} - x^{(k)})$
11: **end if**
12: $k++$
13: **end while**
14: return $\underset{s \in \{s^{(1)}, \ldots, s^{(k-1)}\}}{\text{argmin}} g(s)$

Note k_{end} the last iteration of DFW algorithm.

The main idea of our approach is that we use the classic Frank-Wolfe algorithm on the convex hull Conv(X), and we exploit the fact that $s^{(k)}$ belongs to X thanks to **(A2)**, which means that it is a feasible solution. When Algorithm 2 stops, $x^{(k)}$ is close to the optimal solution x^* of the relaxed problem in the sense that $g(x^{(k)}) - g(x^*) \leq O(\frac{1}{k})$.
Adding to this that $s^{(k)}$ (defined in Algorithm 2, Line 8) is a minimizer of the linear approximation of g in the neighbourhood of $x^{(k)}$, and since $x^{(k)}$ tends to minimize g in Conv(X), this leads us to think that $\text{argmin}_{s \in \{s^{(1)}, \ldots, s^{(k_{\text{end}})}\}} g(s)$ is the best choice to minimize g in X. We choose the line search step $\gamma^{(k)}$ (defined in Algorithm 2, Line 9) because it guarantees that $g(x^{(k)})$ decreases at each iteration. The stopping criteria has been chosen as the convergence of the relaxed problem.

VI. Numerical results

This section is dedicated to illustrate the results of Algorithm 2. First, we describe the experimental setup. Then we discover the evolution of some interesting metrics along the iterations of the algorithm, and we finally compare the solutions and the performance between DFW and the MISOCP solver of CPLEX as a function of the size of the problem.

A. Experimental setup

To test the algorithm, we take the example of the shortest path problem that can be written in the form (1) and verifies the assumptions **(A1)**, **(A2)** and **(A3)**. We consider a directed graph G with n nodes and m edges, and we would like to find an s — d-path with minimal cost, where the costs associated to the edges can be, e.g., the duration or the distance between the nodes. In this type of problems, we have:

$$X = \{x \in \{0,1\}^m; Ax = b\},$$

where

- A is the incidence matrix (of size $n \times m$),
- $b \in \mathbb{R}^m$ is given by $b_i = \mathbb{1}_{\{i=s\}} - \mathbb{1}_{\{i=d\}}$, $i = 1, \ldots, m$,
- s being the source node and d the destination node.

We choose to take grid graphs with L rows and L columns so that $n = L^2$ and $m = 2L(L-1)$.

In the following numerical illustrations, we take $\mu = (\mu_1, \ldots, \mu_m)$ where μ_i are chosen randomly in $[0, 100]$, $i = 1, \ldots, m$. The random covariance Σ matrix is defined as in [16]. Let us write $\Sigma = P^T D P$ where P is an orthogonal eigenvector matrix and D is the corresponding diagonal eigenvalue matrix. Each of the eigenvalues λ_i, $i = 1, \ldots, m$, is chosen as the square of a random number in $[0, \mu_i]$ and P is a random orthogonal matrix.

For the implementation, we use the `Python` language, with the `Networkx` package for creating and manipulating graphs. To compute the solution $s^{(k)}$ (Line 8 of Algorithm 2), we used an LP minimizer with the LP modeler `PuLP`. In all the results of this paper, we set $\varepsilon = 10^{-6}$, $K = 1000$ and $\Omega = 1$.

B. Behavior of DFW algorithm

To observe the behavior DFW algorithm, we take the grid graph with $L = 34$. We denote, at each iteration k, the best solution so far as:

$$s_{opt}^{(k)} = \underset{s^{(l)} \in \{s^{(1)}, \ldots, s^{(k)}\}}{\text{argmin}} g(s^{(l)}).$$

Note then that $s_{opt}^{(k_{\text{end}})}$ is the heuristic solution proposed by DFW. We show in Figure 1 the evolution of $g(s^{(k)})$ and $g(s_{opt}^{(k)})$ in the 200 first iterations.

In this example, the algorithm gives the same solution as CPLEX at iteration $k = 90$ (Figure 1.b.). We see that, although $s^{(k)}$ alternates all along the iterations (Figure 1.a.), it discovers new optimal values as $g(x^{(k)})$ decreases and gets closer to the optimal solution $g(x^*)$.

C. Performance of the DFW algorithm as a function of L

In order to test the DFW Algorithm, we changed the size of the graph, and we compared with the solutions provided by CPLEX. Experiments show that for small to medium graphs, DFW gives the same solution as CPLEX. Arguably, when L is large, methods based on branch-and-bound are no more efficient, which is observed in graphs corresponding to values of L larger than 40. In fact, due to the increase of

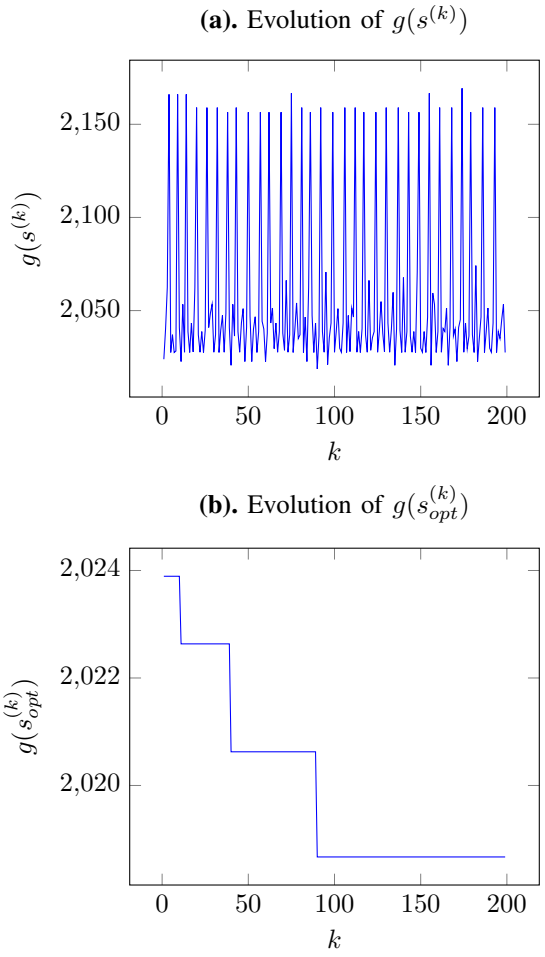

Fig. 1. The evolution of $g(s^{(k)})$ and $g(s^{(k)}_{opt})$ in the 200 first iterations

memory consumption, the console only displays at the end the cost of the best integer solution found so far, rounded off to 4 decimal digits, without the corresponding integer vector, hence we could only compare with this value and thus we could not use CPLEX to obtain a robust solution. We tested graphs with L up to 46, ($n = 2116$ and $m = 4140$), and we observed that the cost of the solution proposed by DFW is the same displayed by CPLEX. This result demonstrates that, even if we are not able to prove that DFW gives the optimal solution, it is a heuristic that is indeed efficient in cases of big graphs, where branch-and-bound methods are no more efficient.

Another interesting fact to mention is that, in more than 97% of the cases, $\varepsilon = 10^{-3}$ is more than enough to obtain the same solution proposed by CPLEX, and in all the experiments, we obtained it at less than 250 iterations.

To sum up the numerical results, the behavior of DFW algorithm is controlled, and it surpassed the MISOCP solver of CPLEX. These numerical findings show that the approach is promising. To understand the importance of the size of graphs taken in the study, one may consider the example of the city of Barcelona. It can be represented with a graph of 1020 nodes and 2522 edges. Another example is Berlin-Mitte-Center which can be represented with 398 nodes and 871 edges. This is to be compared with a grid graph with $40 \times 40 = 1600$ nodes and 3120 edges.

VII. Robust clustering: a second target application

In this section, we consider a robustification of the k-median clustering problem.

Suppose that we are given a set of finite points $P = \{P_1, \dots, P_n\}$. The k-median problem permits to choose the clusters that minimize the sum of the distances between the points $p \in P$ and their cluster centers. It can be expressed in the form of an integer programming problem, as formulated in [23]. The formulation is the following:

$$\begin{aligned} \min_{(z_{pq})_{pq\in\{1,\dots,n\}^2}\in\mathbb{R}^{\{1,\dots,n\}^2}} & \Sigma_{p,q\in\{1,\dots,n\}^2} d(P_p,P_q) z_{pq} \qquad (8)\\ \text{s.t. } & \Sigma_{p\in\{1,\dots,n\}} z_{pq} = 1 \quad \forall q \in \{1,\dots,n\} \\ & z_{pq} \le y_p \quad \forall p,q \in \{1,\dots,n\}^2 \\ & \Sigma_{p\in\{1,\dots,n\}} y_p = k \\ & z_{pq}, y_p \in \{0,1\}, \end{aligned}$$

where $d(P_p, P_q)$ are real positive distances between the points P_p and P_q, y_p indicates whether the point $P_p \in P$ is a cluster center or not, z_{pq} tells us whether $P_q \in P$ is assigned to $P_p \in P$ as center or not. The constraints of (8) assure that each point is assigned to one and only cluster center, that we do not assign a point to another one unless the second is a center and that there exist k centers. In practice, the variables $(z_{pq})_{pq\in\{1,\dots,n\}^2}$ are represented by a $n \times n$ matrix, with $(y_p)_{p\in\{1,\dots,n\}}$ in its the diagonal entries.

In real life applications, the distances between the points defined above are subject to uncertainty. Thus a robust clustering solution seems mandatory. The aim of the following is to motivate the possibility to apply our approach for the k-median clustering, by writing (8) in the formulation (1).

Following (8) and considering the concatenated version z (respectively d) of length n^2 of $(z_{pq})_{pq\in\{1,\dots,n\}^2}$ (respectively $(d(P_p,P_q))_{pq\in\{1,\dots,n\}^2}$), the deterministic k-median problem can be written as

$$\begin{aligned} & \min d^T z \qquad (9)\\ \text{s.t. } & \Sigma_{i=1}^n z_{n(i-1)+j} = 1 \quad \forall j = 1,\dots,n \\ & z_{n(i-1)+j} \le z_{n(i-1)+i} \quad \forall i,j = 1,\dots,n \\ & \Sigma_{i=1}^n z_{n(i-1)+i} = k \\ & z \in \{0,1\}^{n^2} \end{aligned}$$

This problem has the formulation of (1) which can be written as

$$\min_{z\in X} d^T z \qquad (10)$$

with $X \subseteq \{0,1\}^{n^2}$ and where

$$X = \{z \in \{0,1\}^{n^2} | \Sigma_{i=1}^{n} z_{n(i-1)+j} = 1 \quad \forall j = 1, \ldots, n,$$
$$z_{n(i-1)+j} \leq z_{n(i-1)+i}$$
$$\forall i, j = 1, \ldots, n, \Sigma_{i=1}^{n} z_{n(i-1)+i} = k\}.$$

So supposing that the distances between the points are uncertain and making the assumption that d has a multinormal distribution with expectation $\mu \in \mathbb{R}^m$ and covariance matrix $\Sigma \in \mathbb{R}^{m \times m}$, if we follow the development done previously, then the robust clustering problem is reduced to solving the following non-deterministic problem:

$$\min_{z \in X} \mu^T z + \Omega \sqrt{z^T \Sigma z} \tag{11}$$

The formulation done above could permit us to use DFW to solve (11), by taking advantage of the work done in [23], where Awasthi *et al.* studied exact recovery conditions for convex relaxations of the k-median problem. These conditions should serve us for the integrality of the inner iterates of our algorithm.

VIII. Conclusion

This paper proposes a heuristic algorithm that solves the robust counterpart of a general formulation of integer optimization problems with ellipsoidal uncertainty sets. The proposed algorithm is a relaxation-guided version of the Frank-Wolfe algorithm, where we are interested in the optimum of the linear approximation that the algorithm computes at each iteration when relaxing the constraint set in its convex hull. This is legitimate in some problems, such as the shortest path problem, where the computed solutions are feasible solutions for the discrete problem. Numerous numerical experiments have been carried for the robust shortest path problem, and comparisons with the optimal solution given by the second order cone programming solver of CPLEX have been done. Results show that our approach always gives the same solution as CPLEX in the instances where this solver is able to propose one. In addition, the scalability of our algorithm has been approved in problems of large size, where branch-and-bound methods are no more efficient. The approach could be equally applicable on the k-median clustering problem, where the exact recovery conditions for convex relaxations are satisfied. This could have a decisive impact on artificial intelligence and PHM domains concerning data analysis.

Future work will include real applications and experimental studies using real input data, focused on real classification problems encountered in PHM. Moreover, duality-based validation of the approach is ongoing and aims at evaluating the quality of the obtained solutions in the cases where it is no more possible to use a branch-and-bound method.

Acknowledgment

This work has been supported by the EIPHI Graduate school (contract "ANR-17-EURE-0002").

References

[1] R. Gouriveau, K. Medjaher, and N. Zerhouni, *From prognostics and health systems management to predictive maintenance 1: Monitoring and prognostics*. John Wiley & Sons, 2016.

[2] B. Chebel-Morello, J.-M. Nicod, and C. Varnier, *From Prognostics and Health Systems Management to Predictive Maintenance 2: Knowledge, Reliability and Decision*. John Wiley & Sons, 2017.

[3] N. Herr, J.-M. Nicod, and C. Varnier, "Prognostics-based scheduling in a distributed platform: Model, complexity and resolution," in *2014 IEEE International Conference on Automation Science and Engineering (CASE)*. IEEE, 2014, pp. 1054–1059.

[4] S. Chrétien, N. Herr, J.-M. Nicod, and C. Varnier, "A post-prognostics decision approach to optimize the commitment of fuel cell systems in stationary applications," in *2015 IEEE Conference on Prognostics and Health Management (PHM)*. IEEE, 2015, pp. 1–7.

[5] N. Herr, J.-M. Nicod, and C. Varnier, "Prognostic decision making to extend a platform useful life under service constraint," in *2014 International Conference on Prognostics and Health Management*. IEEE, 2014, pp. 1–11.

[6] N. Komodakis, N. Paragios, and G. Tziritas, "Clustering via lp-based stabilities," in *Advances in neural information processing systems*, 2009, pp. 865–872.

[7] C. Buchheim and J. Kurtz, "Robust combinatorial optimization under convex and discrete cost uncertainty," *EURO Journal on Computational Optimization*, vol. 6, no. 3, pp. 211–238, 2018.

[8] "IBM academic portal," https://www.ibm.com/academic.

[9] P. Kouvelis and G. Yu, *Robust discrete optimization and its applications*, OCLC: 854966265.

[10] A. Ben-Tal, A. Goryashko, E. Guslitzer, and A. Nemirovski, "Adjustable robust solutions of uncertain linear programs," *Mathematical Programming*, vol. 99, no. 2, pp. 351–376, 2004.

[11] G. A. Hanasusanto, D. Kuhn, and W. Wiesemann, "K-adaptability in two-stage distributionally robust binary programming," *Operations Research Letters*, vol. 44, no. 1, pp. 6–11, 2016.

[12] M. Fischetti and M. Monaci, "Light robustness," in *Robust and online large-scale optimization*. Springer, 2009, pp. 61–84.

[13] D. Bertsimas and M. Sim, "Robust discrete optimization and network flows," *Mathematical programming*, vol. 98, no. 1-3, pp. 49–71, 2003.

[14] F. Baumann, C. Buchheim, and A. Ilyina, "A lagrangean decomposition approach for robust combinatorial optimization," in *Technical report*. Optimization Online, 2014.

[15] D. Bertsimas and M. Sim, "Robust discrete optimization under ellipsoidal uncertainty sets," 2004.

[16] A. Ilyina, "Combinatorial optimization under ellipsoidal uncertainty," Ph.D. dissertation, Technische Universität Dortmund, 2017.

[17] C. Buchheim, M. De Santis, F. Rinaldi, and L. Trieu, "A frank–wolfe based branch-and-bound algorithm for mean-risk optimization," *Journal of Global Optimization*, vol. 70, no. 3, pp. 625–644, 2018.

[18] H. Markowitz, "Portfolio selection," *The journal of finance*, vol. 7, no. 1, pp. 77–91, 1952.

[19] F. Alizadeh and D. Goldfarb, "Second-order cone programming," *Mathematical programming*, vol. 95, no. 1, pp. 3–51, 2003.

[20] M. Frank and P. Wolfe, "An algorithm for quadratic programming," *Naval research logistics quarterly*, vol. 3, no. 1-2, pp. 95–110, 1956.

[21] J. Lee, *A first course in combinatorial optimization*. Cambridge University Press, 2004, vol. 36.

[22] M. Jaggi, "Revisiting frank-wolfe: Projection-free sparse convex optimization." in *ICML (1)*, 2013, pp. 427–435.

[23] P. Awasthi, A. S. Bandeira, M. Charikar, R. Krishnaswamy, S. Villar, and R. Ward, "Relax, no need to round: Integrality of clustering formulations," in *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*. ACM, 2015, pp. 191–200.