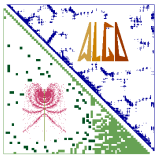
CERFACS

Centre Européen de Recherche et de Formation Avancée en Calcul Scientifique

Variable Neighborhood Search for Robust Optimization and Applications to Aerodynamics

A. Mucherino, M. Fuchs, X. Vasseur, S. Gratton

Technical Report TR-PA-11-25



Publications of the Parallel Algorithms Team

http://www.cerfacs.fr/algor/publications/

Variable Neighborhood Search for Robust Optimization and Applications to Aerodynamics

A. Mucherino[1], M. Fuchs[1], X. Vasseur[1], and S. Gratton[1,2]

[1] CERFACS, Toulouse, France,
{mucherino,martin.fuchs,xavier.vasseur}@cerfacs.fr
[2] INPT-IRIT, University of Toulouse, Toulouse, France,
serge.gratton@enseeiht.fr

Abstract. Many real-life applications lead to the definition of robust optimization problems where the objective function is a black box. This may be due, for example, to the fact that the objective function is evaluated through computer simulations, and that some parameters are uncertain. When this is the case, existing algorithms for optimization are not able to provide good-quality solutions in general. We propose a heuristic algorithm for solving black box robust optimization problems based on the minimax formulation of the problem. We also apply this algorithm for the solution of a wing shape optimization where the objective function is a computationally expensive black box. Preliminary computational experiments are reported.

1 Introduction

Global optimization aims at finding one or more solutions to a problem for which the objective function value is optimized and all constraints, if any, are satisfied. Optimization problems in many real-life applications are NP-hard, because of the complexity of the objective function and/or the dimensionality of the solution set. Moreover, in some applications, this complexity is increased by the necessity of considering some uncertain parameters. If such parameters can range in sufficiently large domains, the solution of the optimization problem, where only one possible set of values for the parameters is considered, may not provide any useful information to the final user.

In real-life applications, uncertainty may concern prices, demands, traveling conditions, working time, and so on [14]. Robust optimization is a way to manage these uncertain parameters in optimization problems [1]. The main difficulty is that the uncertain domain may be quite large and it could lead to the definition of a large-scale optimization problem. In recent years, the scientific community has been giving a lot of attention to algorithms and methods for robust optimization.

As a consequence, different approaches for dealing with uncertainty have been proposed over time [3, 9, 14]. In this work, we consider the approach in which the uncertain parameters are transformed in decision variables. This might seem a simple modification in the problem formulation, but it actually leads to a series of important consequences. First of all, the introduced variables (in the following we will refer to these variables as *uncertain variables*) cannot play the same role of the original decision variables, in the sense that these two kinds of variables do not have to be optimized in the same way.

Generally speaking, when this approach is considered, the uncertain variables must be optimized so that they correspond to the *worst-case scenario* they are able to describe. In this way, the original variables are optimized under the situation in which the worst values for the uncertain variables are considered. If the found solution is optimal for the worst-case scenario, it *should* still be acceptable when the uncertain variables have values different from the worst ones. As remarked in [3], this approach is only able to optimize the objective

function under the worst-case scenario and is not able to give a general good solution for the other values for the uncertain parameters. However, a trade-off between worst-case and expected values could be used, leading to multicriteria optimization. Alternatively one could propagate full probability distributions or more generally cloud models [4].

Let us suppose that the following global optimization problem needs to be solved:

$$\min_{x \in B} f(x, y),$$

where x is a vector of the decision variables, $B \subseteq \Re^{d_x}$, y is a vector of the uncertain parameters, $y \in C \subseteq \Re^{d_y}$, and $f : B \times C \to \Re$. We remark that the set C, in some applications, may depend on the variables x.

If one set of values for the parameters y is chosen, the problem can be solved by employing existing algorithms for global optimization. If we need instead to optimize the objective function under the worst-case scenario that the uncertain variables are able to give, the problem can be rewritten as the following bilevel program:

$$\begin{aligned} &\min_{x \in B} f(x, t) \\ &\text{s.t.} \quad t = \arg\max_{y \in C} f(x, y). \end{aligned} \tag{1}$$

Note that the inner and the outer problem of this bilevel program have both the same objective function $f(x, y)$, but the actual variables are the x's in the outer problem, and the y's in the inner problem. Since the original problem is a minimization problem, the worst-case scenario given by the uncertain variables y can be found by maximizing $f(x, y)$. It is important to note that there is a strong dependence between the two problems, so that they cannot simply be solved one after another (e.g. first solve the inner problem and then solve the outer one), but rather simultaneously. This makes the solution of bilevel programs quite difficult. This kind of problem is also known in the scientific community as the *minimax* optimization problem. We remark that minimax problems with convex-concave payoff and convex constraints can be reduced to single optimization problems [2].

It is important to remark that these optimization problems may become even harder if the mathematical expression of the objective function is not known. In many real-life applications, indeed, the evaluation of the objective function can only be performed through a computer simulation: in other words, $f(x, y)$ is a *black box*. In such a case, methods for optimization which need information on the gradient or even the Hessian matrix of $f(x, y)$ cannot be employed, because this information is usually not available or not sufficiently reliable. Therefore, heuristic algorithms are usually used for solving this class of robust optimization problems. Some examples are given in [8, 12, 15].

In this paper, we propose a heuristic specifically designed for solving bilevel programs and we apply this algorithm for the solution of minimax problems arising in the context of robust optimization. The Variable Neighborhood Search (VNS) [6, 10] is a meta-heuristic search based on the idea of exploring larger and larger neighbors of the current solution in order to discover better solutions. Our

heuristic implements two VNS's, one inserted into another. This idea is not new: it has been already employed for improving the performances of the algorithm for single optimization problems (see for example [10]). Our idea is instead to have the two VNS's work simultaneously on the outer and on the inner problem of the bilevel program. While one VNS can work for optimizing the decision variables x, the other one can work for finding the worst-case scenario given by the uncertain variables y. We will refer to this heuristic algorithm as the *bilevel* VNS.

We employ our bilevel VNS algorithm to solve an aerodynamic shape optimization problem. The considered problem consists in finding the optimal wing shape for a certain class of aircraft. Naturally, many parameters are uncertain in this application, because flight conditions can change during time, such as the flying speed. Moreover, the objective function can only be evaluated by a computer simulation. Preliminary computational experiments are presented where our bilevel VNS is used for solving this problem.

The paper is organized as follows. In Section 2 we give a description of the bilevel VNS algorithm for solving minimax optimization problems. Two examples of robust optimization problems are considered in Section 3. The first one is a scheduling problem for which the objective function is actually not a black box: we consider this problem with the aim of testing our algorithm on a problem for which the solution can be estimated (Section 3.1). The second problem is the one of finding the optimal shape of aircraft's wings in the worst-case scenario given by the flight operating conditions (Section 3.2). In this case, the objective function is actually evaluated through a computer simulation that lasts several minutes for the easier instances. Final remarks and conclusions are given in Section 4.

2 The bilevel VNS

The meta-heuristic Variable Neighborhood Search (VNS) [6, 10] is one of the most successful heuristics for global optimization. It is based on the idea of exploring small neighbors of currently known solutions, which may increase in size only if no better solutions can be found in the current neighbors. VNS makes usually use of local search algorithms for reducing the search to local minima only. In this way, a path of local optima is defined, that may lead to the global optimum of the considered problem.

The VNS is currently widely employed for solving problems arising in various applications. In some very recent works, the VNS has been used, for example, for solving the p-median clustering problem [5], feature selection problems in data mining [11], and routing [7] and task scheduling [16] problems.

When the objective function of the considered optimization problem is a black box (or not easily differentiable), at each iteration of the VNS, the local search can be replaced by another execution of the VNS. In this case, the algorithm is usually referred to as double VNS, where two VNS's are inserted one into another. This is done in order to improve the quality of the found solutions.

In practice, even if the second VNS does not represent a valid alternative to a (deterministic) local search, it can help in escaping local minima and in finding solutions closer to the optimal ones.

In this work, we rather consider two VNS's for solving simultaneously both the inner and the outer problem of a bilevel program. The basic idea is to have only one VNS working on the inner problem, while the second VNS works on the outer problem. No local searches (neither deterministic ones nor heuristic ones) are employed for reducing the search to local minima only. For this reason, our algorithm can only provide solutions that are relatively close to the optimal ones.

This setup may seem unambitious. However, our interest is not in finding very precise solutions to the problem, but rather just in finding some good approximations for such solutions. This task is indeed already quite difficult to perform for the robust optimization problems discussed in the Introduction. We suppose that the objective function evaluations are very expensive, and therefore the number of such evaluations must be limited in order to perform experiments lasting a reasonable amount of time. Any local search algorithm applied at each iteration of any of the two VNS's may increase drastically the necessary number of function evaluations.

Algorithm 1 is a sketch of our bilevel VNS heuristic for bilevel programs (1). At the beginning, random values for both variables x and y can be chosen so that the corresponding constraints, if any, are satisfied. Thereafter, the first VNS starts. Its execution is controlled by a while loop which stops when ε_y gets larger than ε_{max}. ε_y represents the radius of the current neighbor of y_{best} where solutions to the problem can be picked randomly. It can only range in the interval $[\varepsilon_{min}, \varepsilon_{max}]$. When a better solution is found, ε_y is set back to ε_{min}, otherwise it is modified and it may be increased in value after a certain number of iterations without any improvement.

The first VNS considers the uncertain variables y. Therefore, for each improved solution of the inner problem of the bilevel program, a full execution of the second VNS, on the decision variables x, is performed. The two VNS's behave exactly in the same way. The only difference is that, while the first VNS works on the variables y and tries to maximize the objective function, the second VNS works on the variables x and tries to minimize the function value. In robust optimization applications, this is the reflection of the fact that the variables y need to be optimized in order to represent the worst-case scenario related to the problem. If $f(x, y)$ were known to have certain properties, such as concavity in C and convexity in B, a saddle point would exist [13], and the algorithm would attempt to find it.

3 Development and testing

We implemented our bilevel VNS heuristic for bilevel programs in Python. We chose this programming language because part of the code for evaluating the objective function of the problem described in Section 3.2 was already written in this language. The experiments have been performed on one core of an Intel

Algorithm 1 A bilevel VNS heuristic for bilevel programs.

let $iter_x = 0$; let $iter_y = 0$;
let $\varepsilon_x = \varepsilon_{min}$; let $\varepsilon_y = \varepsilon_{min}$;
randomly generate $x \in B$;
randomly generate $y \in C$;
let $x^{best} = x$; let $y^{best} = y$; let $f^{best} = f(x, y)$;
while $(\varepsilon_y \leq \varepsilon_{max})$ **do**
  let $iter_y = iter_y + 1$;
  randomly pick $y \in C$ from a neighborhood of y^{best} with radius ε_y;
  if $(f(x^{best}, y) > f^{best})$ **then**
    let $\varepsilon_y = \varepsilon_{min}$;
    let $y^{best} = y$; let $f^{best} = f(x^{best}, y)$;
    while $(\varepsilon_x \leq \varepsilon_{max})$ **do**
      let $iter_x = iter_x + 1$;
      randomly pick $x \in B$ from a neighborhood of x^{best} with radius ε_x;
      if $(f(x, y^{best}) < f^{best})$ **then**
        let $\varepsilon_x = \varepsilon_{min}$;
        let $x^{best} = x$; let $f^{best} = f(x, y^{best})$;
      else
        modify ε_x;
      end if
    end while
  else
    modify ε_y;
  end if
end while

Xeon 8 CPU E5630 @ 2.53 GHz with 6GB RAM, running Linux. For the experiments in Section 3.2, each function evaluation has actually been performed on a different machine, where the necessary software was installed. More details are given in Section 3.2.

3.1 CPU scheduling

This is the problem of efficiently scheduling CPUs of a parallel machine [8]. We consider this problem with the aim of testing the performances of our bilevel VNS algorithm. Deterministic or hybrid methods may also be used for its solution.

Let us suppose there is a parallel machine composed by m CPUs having exactly the same properties. Let us suppose we have a set of n jobs to be assigned to the CPUs of this parallel computer. For each job $j \in J$, where $J = \{1, 2, \ldots, n\}$, and for each CPU $k \in K$, where $K = \{1, 2, \ldots, m\}$, we can define the binary decision variable x_{jk}, which is 1 if the j^{th} job is assigned to the k^{th} CPU, and it is 0 otherwise. For each job, there is uncertainty on the total processing time, and therefore a lower bound p_j and an upper bound q_j are available for each job

instance name	m	n	nsg	nna	$iter_x$	$iter_y$	$f(x,y)$
test1	2	5	2	4	16	8	5.43
test1	2	5	4	6	70	25	6.28
test1	2	5	6	10	2137	749	6.12
test2	4	9	4	6	105	42	4.81
test2	4	9	6	10	319	116	6.65
test2	4	9	8	12	437	263	6.20

Table 1. Experiments on the CPU scheduling problem.

j. The objective function for our minimax problem is therefore:

$$f(x,y) = \max_{k \in K} \left\{ \sum_{j=1}^{n} x_{jk} y_j \right\},$$

where each uncertain variable y_j is constrained between p_j and q_j, and the following constraint must also be satisfied for each job j:

$$\sum_{k \in K} x_{jk} = 1,$$

because every job needs to be assigned to exactly one CPU. This constraint defines the set B related to the variables x, whereas C is a box defined by the bounds p_j and q_j for each job $j \in J$.

Table 1 shows some computational experiments. Instances of the problem have been created for different choices of n and m. For all instances, $p_j = 1$ and $q_j = 2$ for each job $j \in J$. In this way, we can compute a lower bound on the objective function value when the variables y represent the worst-case scenario, which corresponds to the case $y_j = q_j$ for all $j \in J$. In the table, nsg is the maximum number of solutions that are randomly generated in our bilevel VNS before increasing in value ε_x or ε_y. nna is the total number of neighbors which are allowed during the execution of the algorithm: as a consequence, only nna values in $[\varepsilon_{min}, \varepsilon_{max}]$ can be chosen in both VNS's. Finally, for each experiment, $iter_x$ and $iter_y$ represent the final number of performed iterations in the two VNS's. Therefore, the total number of function evaluations is $iter_x + iter_y$.

For both instances `test1` and `test2`, the lower bound lb on the objective function value is 6 when the variables y represent the worst-case scenario, because at least 3 jobs must be assigned to one CPU in both cases. For the instance `test1`, the found solutions are relatively close to lb in all experiments. In the first experiment, the algorithm was not able to find the worst-case scenario, because $f(x,y)$ is smaller than lb. In the last two experiments, instead, *worse* scenarios were identified, and solutions with $f(x,y)$ close to lb were also found. Similar results are obtained for the instance `test2`. Each experiment took no more than 10 seconds of CPU time.

The aim of these experiments was to check the capability of the algorithm to find solutions that are close to the optimal ones by using only a few function

evaluations, rather than comparing it to existing algorithms. In the most expensive experiment in Table 1, for example, the number of function evaluations is almost 3000 (the sum of iterations for the two VNS's). Therefore, if we suppose that the objective function is a black box whose execution requires 5 minutes, the same result can be obtained in about 10 days of CPU time.

3.2 Wing shape optimization

We consider in this section an important problem arising in aerodynamics, which is the one of finding a robust shape for aircraft's wings. A classic approach to this problem, without taking uncertainties into account, leads to a good decision support for engineers, but to a shape which completely lacks robustness. Indeed, slightly adverse conditions of some uncertain parameters may cause the solution to perform significantly worse than expected.

This class of optimization problems typically features black box objective functions that compute, e.g., the drag or the lift of the wing depending on its shape via Computational Fluid Dynamics (CFD) simulations [3]. The extra computational effort to account for robustness amounts to extra objective function evaluations. Here it is crucial to use only very few evaluations in each level as the total budget of evaluations in the optimization phase is very limited. Moreover, the outer level of the bilevel problem is likely to be strongly nonsmooth. Therefore, existing optimization techniques cannot be used to solve this problem.

In this application, each black box call includes the generation of a finite volume mesh associated with the wing shape chosen, and the CFD calculations of the drag and lift properties of the wing, based on either Euler or Navier-Stokes flow. The input of the black box is the specification of the shape geometry and a set of parameters subject to uncertainty (they are usually 5 in the considered instances). The geometry is specified using the concept of Free-Form Deformation (FFD) [3]. Intuitively that is a placement of a given number of *bumps* at given positions on a reference wing. For each bump three variables are introduced: amplitude, position, expansion. Therefore, in general, the dimension of the outer problem of the bilevel program is

$$d_x = 3 \times \text{number of bumps},$$

whereas the inner problem has dimension $d_y = 5$. The sets B and C are two boxes defined by the corresponding bound constraints on the two variables x and y, respectively. The output of the black box is the pressure drag coefficient of the selected wing shape plus a penalty term for the case that the lift of the wing is not sufficiently large. The objective is to minimize this output value. The implementation is a Python interface accounting for massively parallel architectures used to speed up CFD simulation runs. It is currently installed on a machine belonging to the company `Airbus`. Therefore, solver and black box are in practice executed by different computers.

Table 2 shows some preliminary experiments. We consider an instance where the wing shape is represented by using one bump, and the CFD calculations are

instance name	bumps	nsg	nna	$iter_x$	$iter_y$	$f(x,y)$
Euler1	1	2	2	18	68	498.38
Euler1	1	3	3	25	52	688.71
Euler1	1	4	4	23	37	442.98

Table 2. Experiments on the wing shape optimization problem.

based on the Euler flow. As a consequence, $d_x = 3$ and $d_y = 5$. In the second experiment, the obtained value for $f(x, y)$ is worse with respect to the one found in the first experiment, probably because the algorithm was able to find a *worse* scenario. Finally, in the third experiment, the algorithm was able to identify a solution having a smaller objective function value.

4 Conclusions

We proposed a bilevel VNS for robust optimization. This heuristic algorithm is able to manage minimax optimization problems where the objective function is represented by a computationally expensive black box. The algorithm is based on the well-known meta-heuristic algorithm Variable Neighborhood Search (VNS), and on the idea of inserting two VNS's one into another, in order to solve simultaneously the inner and the outer problem of the bilevel program. Some preliminary experiments show promising results on an academic problem and a real-life example. The algorithm is able to identify good preliminary solutions for the test cases even if the total number of function evaluations is very limited. However, we need to better understand the mathematical properties of the considered problems, and perform a wider testing of the algorithm.

Acknowledgments

The authors gratefully acknowledge partial support of the *Fondation de Recherche pour l'Aéronautique et l'Espace* (FRAE) in the framework of the MEMORIA project. We also would like to thank Matthieu Meaux, Marc Montagnac, Melodie Mouffe and Anke Troeltzsch (CERFACS) for the fruitful discussions.

References

1. A. Ben-Tal, L. El Ghaoui, and A. Nemirovski. *Robust Optimization*. Princeton University Press, Princeton and Oxford, 2009.
2. M.D. Canon. Monoextremal representations of a class of minimax problems. *Management Science*, 15(5):228–238, 1969.
3. R. Duvigneau. Aerodynamic shape optimization with uncertain operating conditions using metamodels. Technical Report INRIA/RR–6143–FR+ENG, INRIA, 2007.

4. M. Fuchs and A. Neumaier. Potential based clouds in robust design optimization. *Journal of Statistical Theory and Practice*, 3(1):225–238, 2008.
5. P. Hansen, J. Brimberg, D. Urosevic, and N. Mladenovic. Solving large p-median clustering problems by primaldual variable neighborhood search. *Data Mining and Knowledge Discovery*, 19(3):351–375, 2009.
6. P. Hansen and N. Mladenovic. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3):449–467, 2001.
7. V.C. Hemmelmayr, K.F. Doerner, and R.F. Hartl. A variable neighborhood search heuristic for periodic routing problems. *European Journal of Operational Research*, 195(3):791–802, 2009.
8. J.W. Herrmann. A genetic algorithm for minimax optimization problems. In *Congress on Evolutionary Computation (CEC99)*, pages 1–5, Washington, DC, USA, 1999. IEEE conference proceedings.
9. L. Huyse and R.M. Lewis. Aerodynamic shape optimization of twodimensional airfoils under uncertain operating conditions. Technical Report 2001-1, NASA Langley Research Center Hampton, VA, USA, 2001.
10. M. Mladenovic and P. Hansen. Variable neighborhood search. *Computers and Operations Research*, 24:1097–1100, 1997.
11. A. Mucherino and A. Urtubia. Consistent biclustering and applications to agriculture. In *Proceedings of the Industrial Conference on Data Mining (ICDM10), Workshop on Data Mining and Agriculture (DMA10)*, IbaI Conference Proceedings, pages 105–113, Berlin, Germany, 2010. Springer.
12. T. Rogalsky, R.W. Derksen, and S. Kocabiyik. Differential evolution in aerodynamic optimization. In *In: Proceedings of the 46th Annual Conference of the Canadian Aeronautics and Space Institute*, pages 29–36, 1999.
13. B. Rustem and M.A. Howe. *Algorithms for Worst-case Design with Applications to Risk Management*. Princeton University Press, 2001.
14. N.V. Sahinidis. Optimization under uncertainty: state-of-the-art and opportunities. *Computers and Chemical Engineering*, 28:971–983, 2004.
15. A. Vicini and D. Quagliarella. Airfoil and wing design through hybrid optimization strategies. *American Institute of Aerodynamics and Astronautics*, 37(5), 1999.
16. Y. Wen, H. Xu, and J. Yang. A heuristic-based hybrid genetic-variable neighborhood search algorithm for task scheduling in heterogeneous multiprocessor system. *Information Sciences*, 181(3):567–581, 2011.