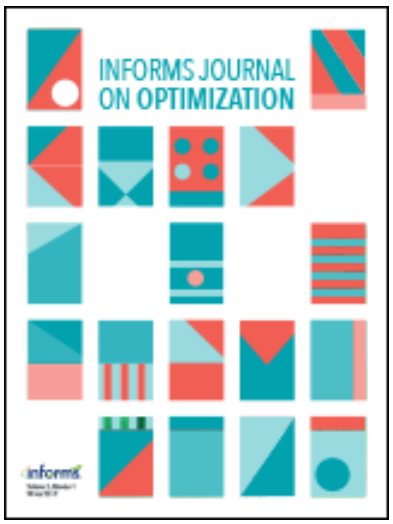


INFORMS Journal on Optimization

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Separable Convex Optimization with Nested Lower and Upper Constraints

Thibaut Vidal, Daniel Gribel, Patrick Jaillet

To cite this article:
Thibaut Vidal, Daniel Gribel, Patrick Jaillet (2019) Separable Convex Optimization with Nested Lower and Upper Constraints. INFORMS Journal on Optimization 1(1):71-90. https://doi.org/10.1287/ijoo.2018.0004

http://pubsonline.informs.org/journal/ijoo/

INFORMS JOURNAL ON OPTIMIZATION
Vol. 1, No. 1, Winter 2019, pp. 71–90
ISSN 2575-1484 (print), ISSN 2575-1492 (online)

Separable Convex Optimization with Nested Lower and Upper Constraints

Thibaut Vidal,[a] Daniel Gribel,[a] Patrick Jaillet[b]

[a] Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro 22451-900, Brazil; [b] Department of Electrical Engineering and Computer Science, Laboratory for Information and Decision Systems, Operations Research Center, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139

Contact: vidalt@inf.puc-rio.br, http://orcid.org/0000-0001-5183-8485 (TV); dgribel@inf.puc-rio.br, https://orcid.org/0000-0002-9850-7162 (DG); jaillet@mit.edu, http://orcid.org/0000-0002-8585-6566 (PJ)

Received: November 2, 2017
Revised: June 19, 2018
Accepted: August 4, 2018
Published Online in Articles in Advance: October 24, 2018

https://doi.org/10.1287/ijoo.2018.0004

Abstract. We study a convex resource allocation problem in which lower and upper bounds are imposed on partial sums of allocations. This model is linked to a large range of applications, including production planning, speed optimization, stratified sampling, support vector machines, portfolio management, and telecommunications. We propose an efficient gradient-free divide-and-conquer algorithm, which uses monotonicity arguments to generate valid bounds from the recursive calls, and eliminate linking constraints based on the information from subproblems. This algorithm does not need strict convexity or differentiability. It produces an ϵ-approximate solution for the continuous problem in $\mathcal{O}(n \log m \log \frac{nB}{\epsilon})$ time and an integer solution in $\mathcal{O}(n \log m \log B)$ time, where n is the number of decision variables, m is the number of constraints, and B is the resource bound. A complexity of $\mathcal{O}(n \log m)$ is also achieved for the linear and quadratic cases. These are the best complexities known to date for this important problem class. Our experimental analyses confirm the good performance of the method, which produces optimal solutions for problems with up to 1,000,000 variables in a few seconds. Promising applications to the support vector ordinal regression problem are also investigated.

Funding: This research was partially supported by the National Counsel of Technological and Scientific Development [Grant 308498/2015-1], Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro in Brazil [Grant E-26/203.310/2016], and the Office of Naval Research [Grants N00014-15-1-2083 and N00014-18-1-2122].

Keywords: convex optimization • resource allocation • nested constraints • speed optimization • lot sizing • stratified sampling • machine learning • support vector ordinal regression

1. Introduction

Resource allocation problems involve the distribution of a fixed quantity of a resource (e.g., time, workforce, money, or energy) over a number of tasks to optimize a value function. In its most fundamental form, the simple resource allocation problem (RAP) is formulated as the minimization of a separable objective subject to one linear constraint representing the total resource bound. Despite its apparent simplicity, this model has been the focus of a considerable research effort over the years, with more than 100 articles, as underlined by the surveys of Patriksson (2008), Katoh et al. (2013), and Patriksson and Strömberg (2015). This level of interest arises from its applications in engineering, production and manufacturing, military operations, machine learning, financial economics, and telecommunications among many other areas.

In several applications, a single global resource bound is not sufficient to model partial budget or investment limits, release dates and deadlines, or inventory or workforce limitations. In these situations, the problem must be generalized to include additional constraints over nested sums of the resource variables. This often leads to the model given in Equations (1)–(4), where the sets $J_i \subseteq \{1, \ldots, n\}$ follow a total order such that $J_i \subset J_{i+1}$ for $i \in \{1, \ldots, m-2\}$:

$$\min \ f(\mathbf{x}) = \sum_{i=1}^{n} f_i(x_i) \tag{1}$$

$$\text{s.t.} \ \ a_i \leq \sum_{j \in J_i} x_j \leq b_i \qquad i \in \{1, \ldots, m-1\} \tag{2}$$

$$\sum_{k=1}^{n} x_k = B \tag{3}$$

$$c_i \leq x_i \leq d_i \qquad i \in \{1, \ldots, n\}. \tag{4}$$

This problem involves a separable convex objective subject to lower and upper bounds on nested subsets of the variables (Equation (2)) and a global resource constraint (Equation (3)). Despite being a special case of the former inequalities, the latter constraint is included in the model to emphasize the resource bound B. Reordering the indices of x_i, c_i, d_i, and f_i, we obtain the formulation given by Equations (5)–(8), where $(\sigma[1], \ldots, \sigma[m])$ is a subsequence of $(1, \ldots, n)$:

$$\min\ f(\mathbf{x}) = \sum_{i=1}^{n} f_i(x_i) \tag{5}$$

$$\text{s.t.}\quad a_i \le \sum_{k=1}^{\sigma[i]} x_k \le b_i \qquad i \in \{1, \ldots, m-1\} \tag{6}$$

$$\sum_{k=1}^{n} x_k = B \tag{7}$$

$$c_i \le x_i \le d_i \qquad i \in \{1, \ldots, n\}. \tag{8}$$

We assume that the functions $f_i : [c_i, d_i] \to \Re$ are Lipschitz continuous but not necessarily differentiable or strictly convex, and the coefficients a_i, b_i, c_i, and d_i are integers. To ease the presentation, we define $a_m = b_m = B$, $\sigma[0] = 0$, and $\sigma[m] = n$. We will study this continuous optimization problem as well as its restriction to integer solutions.

We refer to this problem as the resource allocation problem with nested lower and upper constraints (RAP–NC). As highlighted in Section 2, the applications of this model include production and capacity planning (Love 1973), vessel speed optimization (Psaraftis and Kontovas 2013, 2014), machine learning (Chu and Keerthi 2007), portfolio management (Bienstock 1996), telecommunications (D'Amico et al. 2014), and power management (Gerards et al. 2016). Some of these applications involve large data sets with millions of variables, and in other contexts, multiple RAP–NCs must be repeatedly solved (e.g., thousands or millions of times) to produce bounds in a tree search-based algorithm, optimize vessel speeds over candidate routes within a heuristic search for ship routing, or perform projection steps in a subgradient procedure for a nonseparable objective. In these situations, complexity improvements are a determining factor between success and failure.

The literature contains a rich set of studies and algorithms for a closely related problem: the NESTED resource allocation problem, a special case of the RAP–NC in which $a_i = -\infty$ for all $i \in \{1, \ldots, m-1\}$ (or $b_i = \infty$ for all $i \in \{1, \ldots, m-1\}$). With integer variables, NESTED can be solved in $\mathcal{O}(n \log n \log \frac{B}{n})$ time using a scaling algorithm (Hochbaum 1994) and $\mathcal{O}(n \log m \log \frac{B}{n})$ time using divide-and-conquer principles (Vidal et al. 2016). These algorithms, however, are not applicable for joint lower and upper nested constraints, a case that is essential for a large variety of applications (e.g., to model time windows, time-dependent inventory bounds, or investment ranges).

Moreover, in the presence of continuous variables, the notion of computational complexity for convex problems must be carefully defined, because optimal solutions can be irrational and thus, are not representable in a bit-sized computational model. We will use the same conventions as Hochbaum (1994) by measuring the computational complexity of achieving an *ϵ-approximate* solution guaranteed to be located in the solution space no farther than ϵ from an optimal solution. We also assume that an oracle is available to evaluate each function f_i in $\mathcal{O}(1)$ time. When considering such a model of computation, controlling algorithmic approximations can be a hard task. To circumvent this issue, we will use, as in Hochbaum (1994), a proximity theorem to transform a continuous problem into an integer problem scaled by an appropriate factor and to translate the integer solution back to a continuous solution with the desired precision.

The main contributions of this paper are the following.

- We propose an efficient decomposition algorithm for the convex RAP–NC with integer variables. This algorithm is a nontrivial generalization of that of Vidal et al. (2016) and attains the same complexity of $\mathcal{O}(n \log m \log B)$. Based on a proximity theorem from Moriguchi et al. (2011), we extend this algorithm to solve the continuous problem in $\mathcal{O}(n \log m \log \frac{nB}{\epsilon})$ time. These are the best-known complexities, to date, for both problem variants. The complexity depends on the magnitude of $\log(B/\epsilon)$, a dependency that is known to be unavoidable in the arithmetic model of computation for general forms of convex functions (Renegar 1987). Moreover, the algorithm calls only the oracle for the objective function, without need of gradient information, and does not rely on strict convexity or differentiability. Finally, Lipschitz continuity is assumed for convenience in the proofs but is not mandatory, because an alternative proof line based on submodular optimization could be adopted otherwise.
- For the specific case of quadratic functions, with continuous or integer variables, the method runs in $\mathcal{O}(n \log m)$ time, hence extending the short list of quadratic problems known to be solvable in strongly polynomial time. This also resolves an open question from Moriguchi et al. (2011, p. 654): "It is an open question whether there exist $\mathcal{O}(n \log n)$ algorithms for (Nest) with quadratic objective functions."

- We present computational experiments that show the good performance of the method. We compare it with a known algorithm for the linear case and a general purpose separable convex optimization solver for the convex case using benchmark instances derived from three families of applications.
- We finally integrate the proposed algorithm as a projection step in a projected gradient algorithm for the support vector ordinal regression problem, highlighting promising connections with the machine learning literature.

The rest of the paper is organized as follows. In Section 2, we review related works in a wide variety of application domains. In Section 3, we describe the proposed algorithm and prove its correctness. In Section 4, we report our computational experiments, considering linear and convex objectives, as well as support vector ordinal regression problems. Finally, in Section 5, we provide some concluding remarks.

2. Related Literature and Applications

We now review the many applications of the RAP–NC, starting with classical operations research and management science applications and then moving to statistics, machine learning, and telecommunications.

2.1. Resource Allocation

The resource allocation problem (Equations (1), (3), and (4)) has long been studied as a prototypical problem. The fastest known algorithms (Frederickson and Johnson 1982, Hochbaum 1994) reach a complexity of $\mathbb{O}(n \log \frac{B}{n})$ for the integer problem and can be extended to find an ϵ-approximate solution of the continuous problem in $\mathbb{O}(n \log \frac{B}{\epsilon})$ operations. This complexity is known to be optimal in the algebraic tree model (Hochbaum 1994). Other algorithms have been developed for several generalizations of the RAP in which the constraint set forms a polymatroid. In this context, the greedy algorithm is optimal (Federgruen and Groenevelt 1986), albeit only pseudopolynomial, and efficient scaling algorithms can be developed (Hochbaum 1994, Moriguchi et al. 2011). The special case of the integer RAP–NC with $a_i = -\infty$, called NESTED problem, can be solved in $\mathbb{O}(n \log n \log \frac{B}{n})$ time using a scaling algorithm (Hochbaum 1994) or in $\mathbb{O}(n \log m \log \frac{B}{n})$ time using divide-and-conquer principles (Vidal et al. 2016).

2.2. Production Planning

The formulation given by Equations (5)–(8) is also encountered in early literature on production planning over time with inventory and production costs (Wagner and Whitin 1958). One of the models most closely related to our work is that of Love (1973), with time-dependent inventory bounds. The general problem with concave costs (economies of scale) and production capacities is known to be NP hard. The linear or convex model remains polynomial but is more limited in terms of applicability, although convex production costs can occur in the presence of a limited workforce with possible overtime. Two relatively recent articles have proposed polynomial algorithms for the linear problem with time-dependent inventory bounds (Sedeño-Noda et al. 2004, Ahuja and Hochbaum 2008). With upper bounds x_i^{MAX} on the production quantities and time-dependent inventory capacities I_i^{MAX}, the problem can be stated as

$$\min \quad f(\mathbf{x}, \mathbf{I}) = \sum_{i=1}^{n} p_i(x_i) + \sum_{i=1}^{n} \alpha_i I_i \tag{9}$$

$$\text{s.t.} \quad I_i = I_{i-1} + x_i - d_i \qquad i \in \{2, \ldots, n\} \tag{10}$$

$$I_0 = K \tag{11}$$

$$0 \le I_i \le I_i^{\text{MAX}} \qquad i \in \{1, \ldots, n\} \tag{12}$$

$$0 \le x_i \le x_i^{\text{MAX}} \qquad i \in \{1, \ldots, n\}. \tag{13}$$

Then, expressing the inventory variables as a function of the production quantities using $I_i = K + \sum_{k=1}^{i}(x_k - d_k)$ reduces this problem to an RAP–NC:

$$\min \quad f(\mathbf{x}) = \sum_{i=1}^{n} p_i(x_i) + \sum_{i=1}^{n} \alpha_i \left[K + \sum_{k=1}^{i} (x_k - d_k) \right] \tag{14}$$

$$\text{s.t.} \quad \sum_{k=1}^{i} d_k - K \le \sum_{k=1}^{i} x_k \le \sum_{k=1}^{i} d_k + I_i^{\text{MAX}} - K \qquad i \in \{1, \ldots, n\} \tag{15}$$

$$0 \le x_i \le x_i^{\text{MAX}} \qquad i \in \{1, \ldots, n\}. \tag{16}$$

The objective includes production costs and inventory costs, and the nested constraints model the time-dependent inventory limit. The algorithm of Ahuja and Hochbaum (2008) can solve Equations (9)–(13) in $\mathbb{O}(n \log n)$ time via a reduction to a minimum cost network flow problem. The method was extended to deal with possible

backorders. However, this good complexity comes at the price of an advanced *dynamic tree* data structure (Tarjan and Werneck 2009) that is used to keep track of the inventory capacities.

2.3. Workforce Planning

In contrast with the above studies, which involve the production quantities as decision variables, Bellman et al. (1954) study the balancing of workforce capacity (human or technical resources) over a time horizon under hard production constraints. The variable x_i now represents the workforce variation at period i, and the nested constraints impose bounds on the minimum and maximum workforce in certain periods (e.g., to satisfy forecast production demand). The overall objective, to be minimized, is a convex separable cost function representing positive costs for positive or negative variations of the workforce.

2.4. Vessel Speed Optimization

In an effort to reduce fuel consumption and emissions, shipping companies have adopted *slow-steaming* practices, which moderate ship speeds to reduce costs. This line of research has led to several recent contributions on ship speed optimization, aiming to optimize the vessel speed $v_{i-1,i}$ over each trip segment of length $\delta_{i-1,i}$ while respecting a time window constraint $[a_i, b_i]$ at each destination i. Let $f_i(v_{i-1,i})$ be convex functions representing the fuel costs per mile over $(i-1, i)$ as a function of $v_{i-1,i}$, and let t_i be the arrival time at i. The overall speed optimization problem can be formulated as

$$\min \quad f(\mathbf{t}, \mathbf{v}) = \sum_{i=2}^{n} \delta_{i-1,i} f_i(v_{i-1,i}) \tag{17}$$

$$\text{s.t.} \quad a_i \le t_i \le b_i \qquad i \in \{1, \dots, n\} \tag{18}$$

$$t_{i-1} + \frac{\delta_{i-1,i}}{v_{i-1,i}} \le t_i \qquad i \in \{2, \dots, n\} \tag{19}$$

$$v_{min} \le v_{i-1,i} \le v_{max} \qquad i \in \{2, \dots, n\}. \tag{20}$$

Recent work has considered a constant fuel-speed tradeoff function on each leg (i.e., $f_i = f_j$ for all (i, j)). An $\mathbb{O}(n^2)$ recursive smoothing algorithm (RSA) was proposed by Norstad et al. (2011) and Hvattum et al. (2013) for this case. However, assuming constant fuel-speed over the complete trip is unrealistic, because fuel consumption depends on many varying factors, such as sea condition, weather, current, water depth, and ship load (Psaraftis and Kontovas 2013, 2014). The model can be improved by dividing the trip into smaller segments and considering different functions $f_i \neq f_j$. This more general model falls outside the scope of applicability of RSA.

Let v_i^{opt} be the minimum of each function f_i. With the change of variables $x_1 = t_1$ and $x_i = t_i - t_{i-1}$ for $i \ge 2$, the model can then be reformulated as

$$\min \quad f(\mathbf{x}) = \sum_{i=2}^{n} \delta_{i-1,i} g_i\left(\frac{\delta_{i-1,i}}{x_i}\right) \tag{21}$$

$$\text{s.t.} \quad a_i \le \sum_{k=1}^{i} x_k \le b_i \qquad i \in \{1, \dots, n\} \tag{22}$$

$$\frac{\delta_{i-1,i}}{v_{max}} \le x_i \qquad i \in \{2, \dots, n\}, \tag{23}$$

$$\text{with} \quad g_i(v) = \begin{cases} f_i(v_i^{\text{OPT}}) & \text{if } v \le v_i^{\text{OPT}} \\ f_i(v) & \text{otherwise.} \end{cases} \tag{24}$$

This model is an RAP–NC with separable convex cost. An efficient algorithm for this problem is critical, because a speed optimization algorithm is not often used as a standalone tool but rather, is used as a subprocedure in a route-planning algorithm (Psaraftis and Kontovas 2014). This subprocedure can be called several million times when embedded in a local search on subproblems counting a few hundred variables because of the division of each trip into smaller segments with different sea conditions. Finally, the RAP–NC is also appropriate for variants of vehicle routing problems with emission control (Bektas and Laporte 2011; Kramer et al. 2015a, b) as well as a special case of project crashing for a known critical path (Foldes and Soumis 1993).

2.5. Stratified Sampling

Consider a population of N units divided into subpopulations (*strata*) of $N_1, \dots, N_n$ units such that $N_1 + \cdots + N_n = N$. An optimized stratified sampling method aims to determine the sample size $x_i \in [0, N_i]$ for each stratum to estimate

a characteristic of the population while ensuring a maximum variance level V and minimizing the total sampling cost. Each subpopulation may have a different variance σ_i, and therefore, a sampling plan that is proportional to the size of the subpopulations is frequently suboptimal. The following mathematical model for this sampling design problem was proposed by Neyman (1934) and extended by Srikantan (1963), Hartley (1965), Huddleston et al. (1970), Sanathanan (1971), and others:

$$\min \quad \sum_{i=1}^{n} c_i x_i \tag{25}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} \frac{N_i^2 \sigma_i^2}{N^2} \left(\frac{1}{x_i} - \frac{1}{N_i} \right) \leq V \tag{26}$$

$$0 \leq x_i \leq N_i \qquad i \in \{1, \ldots, n\}. \tag{27}$$

This is a classical RAP formulation. Two extensions of this model are noteworthy in our context. Hartley (1965) and Huddleston et al. (1970) considered multipurpose stratified sampling where more than one characteristic is evaluated while ensuring variance bounds. This leads to several constraints of type (26) and thus, a continuous multidimensional knapsack problem. Sanathanan (1971) considered a hierarchy of strata, with variance bounds for the estimates at each level. This situation occurs, for example, in survey sampling when one seeks an estimate of a characteristic at both the national level (first-stage stratum) and the regional level (second-stage stratum). When two stages are considered, we obtain the additional constraints:

$$\sum_{i \in S_i} \frac{N_i^2 \sigma_i^2}{N^2} \left(\frac{1}{x_i} - \frac{1}{N_i} \right) \leq V_i, \qquad i \in \{1, \ldots, m\}, \tag{28}$$

where the S_i are disjoint sets of strata (i.e., $\bigcup_{i=1}^{m} S_i = \{1, \ldots, n\}$ and $S_i \cap S_j = \emptyset$ for all i, j). The inequalities (28) lead to constraints on the disjoint subsets, giving a resource allocation problem with generalized upper bounds (Hochbaum 1994, Katoh et al. 2013).

2.6. Machine Learning

The support vector machine (SVM) is a supervised learning model, which in its most classical form, seeks to separate a set of samples into two classes according to their labels. This problem is modeled as the search for a separating hyperplane between the projection of the two sample classes into a kernel space of higher dimension, in such a way that the classes are divided by a gap that is as wide as possible and that a penalty for misclassified samples is minimized (Cortes and Vapnik 1995).

As a generalization of the SVM, the support vector ordinal regression aims to find $r - 1$ parallel hyperplanes so as to separate r ordered classes of samples. As reviewed in Gutierrez et al. (2016), various models and algorithms have been proposed in recent years to fulfill this task. In particular, the support vector ordinal regression approach with "explicit constraints on thresholds" (SVOREX) of Chu and Keerthi (2007) obtains a good tradeoff between training speed and generalization capability. A dual formulation of SVOREX is presented in Equations (29)–(33). $\mathcal{K}$ is the kernel function, corresponding to a dot product in the kernel space, and n^j is the number of samples in a class $j \in \{1, \ldots, r\}$. Each dual variable α_i^j takes a nonnull value only when the ith sample of the jth class is active in the definition of the jth hyperplane for $j \in \{1, \ldots, r-1\}$. Similarly, each dual variable α_i^{*j} takes a nonnull value only when the ith sample of the jth class is active in the definition of the $(j-1)$th hyperplane for $j \in \{2, \ldots, r\}$. Additional constraints and variables μ^j impose an order on the hyperplanes. For the sake of simplicity, the dummy variables $\boldsymbol{\alpha}^{*1}$, $\boldsymbol{\alpha}^{r}$, μ^1, and μ^r are defined and should be fixed to zero:

$$\max_{\boldsymbol{\alpha}, \boldsymbol{\alpha}^*, \boldsymbol{\mu}} \sum_{j=1}^{r} \sum_{i=1}^{n^j} (\alpha_i^j + \alpha_i^{*j}) - \frac{1}{2} \sum_{j=1}^{r} \sum_{i=1}^{n^j} \sum_{j'=1}^{r} \sum_{i'=1}^{n^{j'}} (\alpha_i^{*j} - \alpha_i^j)(\alpha_{i'}^{*j'} - \alpha_{i'}^{j'}) \mathcal{K}(x_i^j, x_{i'}^{j'}) \tag{29}$$

$$\text{s.t.} \quad 0 \leq \alpha_i^j \leq C \qquad j \in \{1, \ldots, r\}, i \in \{1, \ldots, n^j\} \tag{30}$$

$$0 \leq \alpha_i^{*j} \leq C \qquad j \in \{1, \ldots, r-1\}, i \in \{1, \ldots, n^j\} \tag{31}$$

$$\sum_{i=1}^{n^j} \alpha_i^j + \mu^j = \sum_{i=1}^{n^{j+1}} \alpha_i^{*j+1} + \mu^{j+1} \qquad j \in \{1, \ldots, r-1\} \tag{32}$$

$$\mu^j \geq 0 \qquad j \in \{1, \ldots, r-1\}. \tag{33}$$

The last two constraints of Equations (32) and (33) can be reformulated to eliminate the μ variables, leading to nested constraints on the variables α and $-\alpha^*$:

$$\sum_{k=1}^{j}\left(\sum_{i=1}^{n^k}\alpha_i^k-\sum_{i=1}^{n^{k+1}}\alpha_i^{*k+1}\right)\geq 0 \qquad j\in\{1,\ldots,r-2\} \tag{34}$$

$$\sum_{k=1}^{r-1}\left(\sum_{i=1}^{n^k}\alpha_i^k-\sum_{i=1}^{n^{k+1}}\alpha_i^{*k+1}\right)=0. \tag{35}$$

Overall, the problem of Equations (29)–(31), (34), and (35) is a nonseparable convex problem over the same constraint polytope as the RAP–NC. Note that the number of nested constraints, corresponding to the number of classes, is usually much smaller than the number of variables, which is proportional to the total number of samples, and thus, $m \ll n$.

The solutions of this formulation are usually sparse, because only a fraction of the samples (support vectors) defines the active constraints and separating hyperplanes. Given this structure and the size of practical applications, modern solution methods rely on decomposition steps, in which a working set of variables is iteratively reoptimized by a method of choice. Such an approach is referred to as *block-coordinate descent* in Bertsekas et al. (2003). The convergence of the algorithm can be guaranteed by including in the working set the variables that most severely violate the Karush-Kuhn-Tucker (KKT) conditions. Chu and Keerthi (2007), in line with the work of Platt (1998), consider a minimal working set with only two variables at each iteration. The advantage is that the subproblem can be solved analytically in this case, the disadvantage is that a large number of working set selections can be needed for convergence, and the KKT condition check and gradient update may become the bottleneck instead of the optimization itself. To better balance the computational effort and reduce the number of decomposition steps, larger working sets could be considered (e.g., as in SVM$^{\text{LIGHT}}$ of Joachims 1999). Still, to be successful, the algorithm must solve each subproblem, here a nonseparable RAP–NC, very efficiently. Such an alternative optimization approach will be investigated in Section 4.3.

2.7. Portfolio Management

The mean variance optimization model of Markowitz (1952) has been refined over the years to integrate a large variety of constraints. In its most classical form, the model aims to maximize expected return while minimizing a risk measure, such as the variance of the return. This problem can be formulated as

$$\left\{\max\sum_{i=1}^{n}x_i\mu_i\,;\ \min\sum_{i=1}^{n}\sum_{j=1}^{n}x_ix_j\sigma_{ij}\right\} \tag{36}$$

$$\text{s.t.}\quad \sum_{i=1}^{n}x_i=1 \tag{37}$$

$$0\leq x_i \qquad i\in\{1,\ldots,n\}, \tag{38}$$

where the x_i variables, $i\in\{1,\ldots,n\}$, represent investments in different assets; μ_i is the expected return of asset i; and σ_{ij} is the covariance between asset i and j. In this model, Equation (37) is used to normalize the total investment, and Equation (38) prevents short selling. The literature on these models is vast, and we refer to the recent surveys of Kolm et al. (2014) and Mansini et al. (2014) for more thorough descriptions. Two additional constraint families, often used in practical portfolio models, are closely linked with the RAP–NC.

- *Class constraints* limit the investment amounts for certain classes of assets or economic sectors. These can result from regulatory requirements, managerial insights, or customer guidelines (see, e.g., Chang et al. 2000, Anagnostopoulos and Mamanis 2010). The assets may also be ranked into different categories (e.g., based on their risk or ecological impact). Imposing investment bounds at each level leads to the nested constraints of Equation (6).
- *Fixed transaction costs*, *minimum transaction levels*, and *cardinality constraints* either impose a fixed price or threshold quantity for any investment in an asset or limit the number of positions on different assets. These constraints usually require the introduction of additional integer variables y_i, taking value one if and only if the asset i is included in the portfolio. This leads to quadratic mixed integer programs (MIPs), for which metaheuristics (Chang et al. 2000, Crama and Schyns 2003) and branch-and-cut methods (Bienstock 1996, Jobst et al. 2001) form the current state-of-the-art. Bienstock (1996) branches on the y_i variables and solve a quadratic resource allocation problem, with additional surrogate constraints in the form of Equation (6), at each node of the search tree. Improved algorithms for the RAP–NC can thus also prove helpful as a methodological building block for more complex portfolio optimization algorithms.

2.8. Telecommunications

Constrained resource allocation problems also have a variety of applications in telecommunications. Mobile signals, for example, can be emitted in different directions with different power levels, but interference between

signals emitted in the same direction reduces communication quality. In this context, a power and direction must be determined for each signal while respecting service quality constraints and minimizing transmission costs. As underlined by Viswanath and Anantharam (2002) and Padakandla and Sundaresan (2009), this problem can be formulated as an instance of the RAP–NC. Given the large size of typical applications, the efficiency of the algorithm is of foremost importance.

A similar model arises for power minimization in multiple-input and multiple-output communication systems as well as in various other applications of optimization to telecommunications (D'Amico et al. 2014). Moreover, the RAP–NC generalizes a family of multilevel *water-filling* problems, which have been the focus of significant research (Palomar and Fonollosa 2005). Other applications include power management on multimedia devices, which is discussed by Huang and Wang (2009) and Gerards et al. (2016). As illustrated by these example applications, the RAP–NC is a prototypical model and an elementary building block for various problems. Therefore, a new algorithmic breakthrough can have considerable impact in many contexts.

3. Proposed Methodology

In this section, we first describe the proposed methodology for the case of continuous variables and then move on to the case with integer variables. We assume that $a_i \le b_i$ for $i \in \{1, \dots, n\}$; otherwise, the problem is trivially infeasible. We will use boldface notation for vectors and normal font for scalars. Let $\mathbf{e}^s$ be the unit vector such that $e_s = 1$ and $e_i = 0$ for $i \neq s$.

3.1. Continuous RAP–NC

The proposed algorithm for the RAP–NC is a divide-and-conquer approach over the indices of the nested constraints. It can be seen as a generalization of the method of Vidal et al. (2016), with some fundamental differences related to the number and the nature of the subproblems. For each range of indices (v, w) considered during the search such that $1 \le v \le w \le m$, it solves four subproblems, RAP–NC$_{v,w}(L, R)$ for $L \in \{a_{v-1}, b_{v-1}\}$ and $R \in \{a_w, b_w\}$, expressed in Equations (39)–(42), obtained by fixing the $(v-1)$th and wth nested constraints to their lower or upper bounds. M is a large number that is defined to be larger than the Lipschitz constant of each function f_i:

$$\text{RAP–NC}_{v,w}(L,R): \quad \min \ \bar{f}(\mathbf{x}) = \sum_{i=\sigma[v-1]+1}^{\sigma[w]} \bar{f}_i(x_i) \tag{39}$$

$$\text{s.t.} \quad a_i - L \le \sum_{k=\sigma[v-1]+1}^{\sigma[i]} x_k \le b_i - L \qquad i \in \{v, \dots, w-1\} \tag{40}$$

$$\sum_{i=\sigma[v-1]+1}^{\sigma[w]} x_i = R - L \tag{41}$$

$$\text{with } \bar{f}_i(x) = \begin{cases} f_i(c_i) + M(c_i - x) & \text{if } x < c_i \\ f_i(x_i) & \text{if } x \in [c_i, d_i] \\ f_i(d_i) + M(x - d_i) & \text{if } x > d_i. \end{cases} \tag{42}$$

To solve these problems when $v < w$, the algorithm relies on known optimal solutions obtained deeper in a recursion over the range (v, u) and $(u+1, w)$, with $u = \lfloor (v+w)/2 \rfloor$. When $v = w$ (at the bottom of the recursion), the RAP–NC$_{v,v}(L, R)$ does not contain any nested constraints from Equation (40) and thus, reduces to a simple RAP. We will refer to this approach as the *monotonic decomposition algorithm* (MDA) MDA(v, w). The original RAP–NC is solved by MDA$(1, m)$, and the maximum depth of the recursion is $\lceil \log m \rceil$, because the binary decomposition is performed over the m nested constraints.

In the formulation given by Equations (39)–(42), note that the bounds $c_i \le x_i \le d_i$ are transferred into the objective via an exact L1 penalty function. This is possible, because the functions f_i satisfy the Lipschitz condition (Theorem 1), and it helps simplify the exposition and proofs.

Theorem 1 (Relaxation-Penalization). *If there exists a solution x of* RAP–NC$_{v,w}(L, R)$ *such that* $\mathbf{c} \le \mathbf{x} \le \mathbf{d}$, *then all optimal solutions of* RAP–NC$_{v,w}(L, R)$ *satisfy* $\mathbf{c} \le \mathbf{x} \le \mathbf{d}$.

Proof. Assume the existence of an optimal solution $\mathbf{x}^*$ of RAP–NC$_{v,w}(L, R)$ with an index $s \in \{\sigma[v-1]+1, \dots, \sigma[w]\}$ such that $x^*_s > d_s$ and a solution $\mathbf{x}$ such that $\mathbf{c} \le \mathbf{x} \le \mathbf{d}$. Because $x^*_s > d_s \ge x_s$ and $\sum_{k=\sigma[v-1]+1}^{\sigma[w]} x^*_k = \sum_{k=\sigma[v-1]+1}^{\sigma[w]} x_k = R - L$, either

$$\sum_{k=\sigma[v-1]+1}^{s} x^*_k > \sum_{k=\sigma[v-1]+1}^{s} x_k \qquad \text{or} \qquad \sum_{k=s}^{\sigma[w]} x^*_k > \sum_{k=s}^{\sigma[w]} x_k. \tag{43}$$

In the first case, define $t = \min\{i \mid i > s \text{ and } \sum_{k=\sigma[v-1]+1}^{i} x_k^* \le \sum_{k=\sigma[v-1]+1}^{i} x_k\}$. Observe that $x_t^* < x_t$, and thus, $d_t - x_t^* > 0$. Moreover, there exists $\Delta > 0$ such that, for each j such that $\sigma[j] \in \{s, \dots, t-1\}$, $\sum_{k=\sigma[v-1]+1}^{\sigma[j]} x_k^* - \Delta > \sum_{k=\sigma[v-1]+1}^{\sigma[j]} x_k \ge a_i$. Defining $\Delta' = \min\{\Delta, d_t - x_t^*, x_s^* - d_s\}$, the solution $\mathbf{x}'^* = \mathbf{x}^* + \Delta'(\mathbf{e}^t - \mathbf{e}^s)$ is feasible and such that $\bar{f}(\mathbf{x}'^*) = \bar{f}(\mathbf{x}^*) + \bar{f}_t(x_t^* + \Delta') - \bar{f}_t(x_t^*) + \bar{f}_s(x_s^* - \Delta') - \bar{f}_s(x_s^*)$. Because of the Lipschitz condition, we have $\bar{f}_t(x_t^* + \Delta') - \bar{f}_t(x_t^*) < M\Delta'$. Moreover, $M\Delta' = \bar{f}_s(x_s^*) - \bar{f}_s(x_s^* - \Delta')$, and thus, $\bar{f}(\mathbf{x}'^*) < \bar{f}(\mathbf{x}^*)$, contradicting the optimality of $\mathbf{x}^*$.

The second case of Equation (43) is analogous. □

The main challenge of the MDA is now to exploit the information gathered at deeper steps of the recursion to solve each RAP–NC efficiently. For this purpose, we introduce Theorem 2, which expresses a monotonicity property of the optimal solutions as a function of the resource bound R. As shown subsequently in Theorem 3, this result allows us to generate tighter bounds on the variables, which supersede the nested constraints of the RAP–NC and allow us to solve all subproblems (at all recursion levels) as simple RAPs.

Theorem 2 (Monotonicity). *Consider three bounds $R^\downarrow \le R \le R^\uparrow$. If $\mathbf{x}^\downarrow$ is an optimal solution of* RAP–NC$_{v,w}(L, R^\downarrow)$ *and $\mathbf{x}^\uparrow$ is an optimal solution of* RAP–NC$_{v,w}(L, R^\uparrow)$ *such that $\mathbf{x}^\downarrow \le \mathbf{x}^\uparrow$, then there exists an optimal solution $\mathbf{x}^*$ of* RAP–NC$_{v,w}(L, R)$ *such that $\mathbf{x}^\downarrow \le \mathbf{x}^* \le \mathbf{x}^\uparrow$.*

Proof. Define $\bar{a}_i = a_i - L$ and $\bar{b}_i = b_i - L$ for $i \in \{v, \dots, w-1\}$ as well as $\bar{a}_w = \bar{b}_w = R - L$. By the KKT conditions (in the presence of a convex objective over a set of linear constraints), if $\mathbf{x}$ is an optimal solution of RAP–NC$_{v,w}(L, R)$, then there exist dual multipliers $(\boldsymbol{\kappa}, \boldsymbol{\lambda})$ such that

$$\Phi_i = \sum_{k \in \{v,\dots,w\} \mid \sigma[k] \ge i} (\kappa_k - \lambda_k) \in \partial \bar{f}_i(x_i) \qquad i \in \{\sigma[v-1]+1, \dots, \sigma[w]\} \tag{44}$$

$$\bar{a}_i \le \sum_{k=\sigma[v-1]+1}^{\sigma[i]} x_k \le \bar{b}_i \qquad i \in \{v, \dots, w\} \tag{45}$$

$$\kappa_i \left(\sum_{k=\sigma[v-1]+1}^{\sigma[i]} x_k - \bar{a}_i \right) = 0, \kappa_i \in \mathfrak{R}^+ \qquad i \in \{v, \dots, w\} \tag{46}$$

$$\lambda_i \left(\bar{b}_i - \sum_{k=\sigma[v-1]+1}^{\sigma[i]} x_k \right) = 0, \lambda_i \in \mathfrak{R}^+ \qquad i \in \{v, \dots, w\}. \tag{47}$$

Note the appearance of the subgradients $\partial \bar{f}_i$ in Equation (44), because the functions $\bar{f}_i$ are not necessarily differentiable. Let $(\boldsymbol{\kappa}^\uparrow, \boldsymbol{\lambda}^\uparrow, \boldsymbol{\Phi}^\uparrow)$ be a set of multipliers associated with the optimal solution $\mathbf{x}^\uparrow$ of RAP–NC$_{v,w}(L, R^\uparrow)$ and $\mathbf{x}$ be an optimal solution of RAP–NC$_{v,w}(L, R)$. Define $\mathcal{S}_{\mathbf{x}}^+ = \{i \mid x_i > x_i^\uparrow\}$, $\mathcal{S}_{\mathbf{x}}^- = \{i \mid x_i < x_i^\downarrow\}$, and $\mathcal{S}_{\mathbf{x}} = \{i \mid x_i^\downarrow \le x_i \le x_i^\uparrow\}$. We will present a construct that generates a sequence of solutions $(\mathbf{x}^k)$ starting from $\mathbf{x}^0 = \mathbf{x}$, such that $|\mathcal{S}_{\mathbf{x}^{K+1}}^+| < |\mathcal{S}_{\mathbf{x}^K}^+|$ and $|\mathcal{S}_{\mathbf{x}^{K+1}}^-| \le |\mathcal{S}_{\mathbf{x}^K}^-|$ as long as $|\mathcal{S}_{\mathbf{x}^K}^+| > 0$, leading by recurrence to a solution $\bar{\mathbf{x}}$ such that $\bar{\mathbf{x}} \le \mathbf{x}^\uparrow$.

If $|\mathcal{S}_{\mathbf{x}^K}^+| > 0$, then there exists $s \in \{\sigma[v-1]+1, \dots, \sigma[w]\}$ such that $x_s^\uparrow < x_s^K$. Let r be the greatest index in $\{\sigma[v-1]+1, \dots, s\}$ such that $\sum_{k=\sigma[v-1]+1}^{r-1} x_k^K \ge \sum_{k=\sigma[v-1]+1}^{r-1} x_k^\uparrow$, and let t be the smallest index in $\{s, \dots, \sigma[w]\}$ such that $\sum_{k=\sigma[v-1]+1}^{t} x_k^K \le \sum_{k=\sigma[v-1]+1}^{t} x_k^\uparrow$.

Because $R^\uparrow - L = \sum_{i=\sigma[v-1]+1}^{\sigma[w]} x_i^\uparrow \ge \sum_{i=\sigma[v-1]+1}^{\sigma[w]} x_i^K = R - L$ and by the definition of r and t, it follows that $\sum_{i=r}^{t} x_i^\uparrow \ge \sum_{i=r}^{t} x_i^K$. Moreover, $r < s \Rightarrow x_r^K < x_r^\uparrow$, and $s < t \Rightarrow x_t^K < x_t^\uparrow$. Finally, note that $r = s = t$ (jointly) is impossible.

- When $r < s$, the following statements are valid. For each j such that $\sigma[j] \in \{r, \dots, s-1\}$, $\bar{a}_j \le \sum_{k=\sigma[v-1]+1}^{\sigma[j]} x_k^K < \sum_{k=\sigma[v-1]+1}^{\sigma[j]} x_K^\uparrow \le \bar{b}_j$ (by the definition of r), and thus, $\kappa_j^\uparrow = \lambda_j^K = 0$. As a consequence, $\Phi_i^K \ge \Phi_{i+1}^K$ and $\Phi_i^\uparrow \le \Phi_{i+1}^\uparrow$ for $i \in \{r, \dots, s-1\}$. The functions $\bar{f}_i$ are convex, and thus, their (Clarke) subgradients are monotone (Rockafellar 1970) (i.e., $\{x_s^\uparrow < x_s^K, \Phi_s^\uparrow \in \partial \bar{f}_s(x_s^\uparrow), \Phi_s^K \in \partial \bar{f}_s(x_s^K)\} \Rightarrow \Phi_s^\uparrow \le \Phi_s^K$). Similarly, we have $\{x_r^\uparrow > x_r^K, \Phi_r^\uparrow \in \partial \bar{f}_r(x_r^\uparrow), \Phi_r^K \in \partial \bar{f}_r(x_r^K)\} \Rightarrow \Phi_r^\uparrow \ge \Phi_r^K$. Combining these relations leads to

$$\Phi_s^K \le \Phi_r^K \le \Phi_r^\uparrow \le \Phi_s^\uparrow \le \Phi_s^K, \tag{48}$$

and thus, there exists $\Psi \in \mathfrak{R}$ such that $\Phi_i^\uparrow = \Phi_i^K = \Psi$ for $i \in \{r, \dots, s\}$.

- When $s < t$, the following statements are valid. For each j such that $\sigma[j] \in \{s, \dots, t-1\}$, $\bar{a}_j \le \sum_{k=\sigma[v-1]+1}^{\sigma[j]} x_K^\uparrow < \sum_{k=\sigma[v-1]+1}^{\sigma[j]} x_k^K \le \bar{b}_j$ (by the definition of t), and thus, $\lambda_j^\uparrow = \kappa_j^K = 0$. As a consequence, $\Phi_i^K \le \Phi_{i+1}^K$ and $\Phi_i^\uparrow \ge \Phi_{i+1}^\uparrow$ for $i \in \{s, \dots, t-1\}$. Furthermore, as before, $x_s^\uparrow < x_s^K$ and $x_t^\uparrow > x_t^K$, and thus, $\Phi_s^\uparrow \le \Phi_s^K$ and $\Phi_t^\uparrow \ge \Phi_t^K$. Combining these relations leads to

$$\Phi_s^K \le \Phi_t^K \le \Phi_t^\uparrow \le \Phi_s^\uparrow \le \Phi_s^K, \tag{49}$$

and thus, there exists $\Psi \in \mathfrak{R}$ such that $\Phi_i^\uparrow = \Phi_i^K = \Psi$ for $i \in \{s, \dots, t\}$.

Overall, $\Phi_i^{\uparrow} = \Phi_i^{K} = \Psi$ for $i \in \{r, \ldots, t\}$, and thus, $\Psi \in \partial \bar{f}_i(x_i^K) \cap \partial \bar{f}_i(x_i^{\uparrow})$ for $i \in \{r, \ldots, t\}$. Define $x_i^{MIN} = \min\{x_i^K, x_i^{\uparrow}\}$ and $x_i^{MAX} = \text{MAX}\{x_i^K, x_i^{\uparrow}\}$. This implies that $\partial \bar{f}_i(x) = \{\Psi\}$ for $x \in (x_i^{MIN}, x_i^{MAX})$, and thus, these functions are affine with identical slope: $\bar{f}_i(x) = \bar{f}_i(x_i^K) + \Psi(x - x_i^K)$ for $x \in [x_i^{MIN}, x_i^{MAX}]$. We can thus transfer value from the variables of the set $\mathcal{S}^+ = \mathcal{S}^+_{\mathbf{x}^K} \cap \{r, \ldots, t\}$ to those of the set $\overline{\mathcal{S}^+} = \{r, \ldots, t\} - \mathcal{S}^+$ via Adjust($[r, \ldots, t], \mathbf{x}^K, \mathbf{x}^{\uparrow}$) (Algorithm 1), leading to a feasible solution $\mathbf{x}^{K+1}$ with the same cost as $\mathbf{x}^K$ (hence, optimal) such that

$$\begin{cases} x_i^{K+1} = x_i^{\uparrow} & \text{for } i \in \mathcal{S}^+ \\ x_i^K \le x_i^{K+1} \le x_i^{\uparrow} & \text{for } i \in \overline{\mathcal{S}^+} \\ x_i^{K+1} = x_i^K & \text{otherwise.} \end{cases}$$

We observe that $|\mathcal{S}^+_{\mathbf{x}^{K+1}}| < |\mathcal{S}^+_{\mathbf{x}^K}|$; moreover, $|\mathcal{S}^-_{\mathbf{x}^{K+1}}| \le |\mathcal{S}^-_{\mathbf{x}^K}|$. By recurrence, repeating the previous transformation leads to a solution $\bar{\mathbf{x}}$ such that $\mathcal{S}^+_{\bar{\mathbf{x}}} = \emptyset$. A similar principle can then be applied to generate a sequence of solutions $(\bar{\mathbf{x}}^K)$, starting from $\bar{\mathbf{x}}^0 = \bar{\mathbf{x}}$, such that $|\mathcal{S}^-_{\bar{\mathbf{x}}^{K+1}}| < |\mathcal{S}^-_{\bar{\mathbf{x}}^K}|$ and $\mathcal{S}^+_{\bar{\mathbf{x}}^{K+1}} = \emptyset$ as long as $|\mathcal{S}^-_{\bar{\mathbf{x}}^K}| > 0$, leading to an optimal solution $\mathbf{x}^*$ such that $\mathbf{x}^{\downarrow} \le \mathbf{x}^* \le \mathbf{x}^{\uparrow}$. □

Algorithm 1 (Adjust($V, \mathbf{x}, \mathbf{x}^{\uparrow}$))

```
Δ ← 0;
for i = V_1, ..., V_|V|, do
    if x_i > x_i^↑, then
        Δ ← Δ + x_i − x_i^↑;
        x_i ← x_i^↑;
for i = V_1, ..., V_|V|, do
    if x_i < x_i^↑, then
        δ = min{x_i^↑ − x_i, Δ};
        x_i = x_i + δ;
        Δ = Δ − δ.
```

Theorem 3 (Variable Bounds). *Let* $\mathbf{x^{La}}$, $\mathbf{x^{Lb}}$, $\mathbf{x^{aR}}$, *and* $\mathbf{x^{bR}}$ *be optimal solutions of* RAP–NC$_{v,u}(L, a_u)$, RAP–NC$_{v,u}(L, b_u)$, RAP–NC$_{u+1,w}(a_u, R)$, *and* RAP–NC$_{u+1,w}(b_u, R)$, *respectively. If* $\mathbf{x^{La}} \le \mathbf{x^{Lb}}$ *and* $\mathbf{x^{bR}} \le \mathbf{x^{aR}}$, *then there exists an optimal solution* $\mathbf{x}^*$ *of* RAP–NC$_{v,w}(L, R)$ *such that*

$$x_i^{La} \le x_i^* \le x_i^{Lb} \quad \text{for } i \in \{\sigma[v-1]+1, \ldots, \sigma[u]\} \text{ and} \tag{50}$$

$$x_i^{bR} \le x_i^* \le x_i^{aR} \quad \text{for } i \in \{\sigma[u]+1, \ldots, \sigma[w]\}. \tag{51}$$

Proof. Let $\mathbf{x}$ be an optimal solution of RAP–NC$_{v,w}(L, R)$. As such, $(x_{\sigma[v-1]+1}, \ldots, x_{\sigma[u]})$ and $(x_{\sigma[u]+1}, \ldots, x_{\sigma[w]})$ must be optimal solutions of RAP–NC$_{v,u}(L, X)$ and RAP–NC$_{u+1,w}(X, R)$ with $X = L + \sum_{i=\sigma[v-1]+1}^{\sigma[u]} x_i$. Because $a_u \le X \le b_u$, there exists an optimal solution $\mathbf{x}^*$ of RAP–NC$_{v,u}(L, X)$ such that $x_i^{La} \le x_i^* \le x_i^{Lb}$ for $i \in \{\sigma[v-1]+1, \ldots, \sigma[u]\}$ via Theorem 2. The other inequality is obtained for $i \in \{\sigma[u]+1, \ldots, \sigma[w]\}$ with the same argument after reindexing the variables downward from $\sigma[w]$ to $\sigma[u]+1$. □

As a consequence of Theorems 2 and 3, the inequalities of Equations (50) and (51) are valid and can be added to the RAP–NC formulation given by Equations (39)–(42). Moreover, we show that these inequalities dominate the nested constraints of Equation (40). Indeed,

$$\begin{aligned} & x_k^{La} \le x_k \le x_k^{Lb} \text{ for } k \in \{\sigma[v-1]+1, \ldots, \sigma[u]\} \text{ and } i \in \{v, \ldots, u\} \\ & \quad \Rightarrow \sum_{k=\sigma[v-1]+1}^{\sigma[i]} x_k^{La} \le \sum_{k=\sigma[v-1]+1}^{\sigma[i]} x_k \le \sum_{k=\sigma[v-1]+1}^{\sigma[i]} x_k^{Lb} \\ & \quad \Rightarrow \bar{a}_i \le \sum_{k=\sigma[v-1]+1}^{\sigma[i]} x_k \le \bar{b}_i \quad \text{and} \end{aligned} \tag{52}$$

$$\begin{aligned} & x_k^{bR} \le x_k \le x_k^{aR} \text{ for } k \in \{\sigma[u]+1, \ldots, \sigma[w]\} \text{ and } i \in \{u, \ldots, w-1\} \\ & \quad \Rightarrow \sum_{k=\sigma[i]+1}^{\sigma[w]} x_k^{bR} \le \sum_{k=\sigma[i]+1}^{\sigma[w]} x_k \le \sum_{k=\sigma[i]+1}^{\sigma[w]} x_k^{aR}. \end{aligned} \tag{53}$$

Moreover, Equations (50) and (51) imply that

$$\sum_{k=\sigma[v-1]+1}^{\sigma[u]} x_k^{Lb} + \sum_{k=\sigma[u]+1}^{\sigma[w]} x_k^{bR} = \sum_{k=\sigma[v-1]+1}^{\sigma[w]} x_k = \sum_{k=\sigma[v-1]+1}^{\sigma[u]} x_k^{La} + \sum_{k=\sigma[u]+1}^{\sigma[w]} x_k^{aR} = R - L, \tag{54}$$

and combining Equations (53) and (54) leads to

$$\begin{aligned} &\Rightarrow \sum_{k=\sigma[v-1]+1}^{\sigma[u]} x_k^{Lb} + \sum_{k=\sigma[u]+1}^{\sigma[i]} x_k^{bR} \geq \sum_{k=\sigma[v-1]+1}^{\sigma[i]} x_k \geq \sum_{k=\sigma[v-1]+1}^{\sigma[u]} x_k^{La} + \sum_{k=\sigma[u]+1}^{\sigma[i]} x_k^{aR} \\ &\Rightarrow \qquad \bar{b}_i \geq \sum_{k=\sigma[v-1]+1}^{\sigma[i]} x_k \geq \bar{a}_i. \end{aligned} \tag{55}$$

Therefore, the nested constraints are superseded at each level of the recursion by the variable bounds obtained from the subproblems. The immediate consequence is a problem simplification: without nested constraints, the formulation reduces to a simple RAP given in Equations (56)–(58), which can be efficiently solved by the algorithm of Frederickson and Johnson (1982) or Hochbaum (1994):

$$\text{RAP}_{v,w}(L, R, \bar{\mathbf{c}}, \bar{\mathbf{d}}): \quad \min \ \bar{f}(\mathbf{x}) = \sum_{i=\sigma[v-1]+1}^{\sigma[w]} \bar{f}_i(x_i) \tag{56}$$

$$\text{s.t.} \quad \sum_{i=\sigma[v-1]+1}^{\sigma[w]} x_i = R - L \tag{57}$$

$$\bar{c}_i \leq x_i \leq \bar{d}_i \qquad i \in \{\sigma[v-1]+1, \ldots, \sigma[w]\}. \tag{58}$$

The pseudocode of the overall decomposition approach is summarized in Algorithm 2.

Algorithm 2 (MDA(v, w))

```
if v = w, then
    (x^aa_{σ[v−1]+1}, ..., x^aa_{σ[v]}) ← RAP_{v,v}(a_{v−1}, a_w, −∞, ∞);
    (x^ab_{σ[v−1]+1}, ..., x^ab_{σ[v]}) ← RAP_{v,v}(a_{v−1}, b_w, −∞, ∞);
    (x^ba_{σ[v−1]+1}, ..., x^ba_{σ[v]}) ← RAP_{v,v}(b_{v−1}, a_w, −∞, ∞);
    (x^bb_{σ[v−1]+1}, ..., x^bb_{σ[v]}) ← RAP_{v,v}(b_{v−1}, b_w, −∞, ∞);
else
    u ← ⌊(v+w)/2⌋;
    MDA(v, u);
    MDA(u + 1, w);
    for (L, R) ∈ {(a, a), (a, b), (b, a), (b, b)}, do
        if x^La ≰ x^Lb, then x^La ← Adjust([σ[v − 1] + 1, ..., σ[u]], x^La, x^Lb);
        for i = σ[v − 1] + 1 to σ[u], do
            [c̄_i, d̄_i] ← [x_i^La, x_i^Lb];
        if x^bR ≰ x^aR, then x^bR ← Adjust([σ[w], ..., σ[u] + 1], x^bR, x^aR);
        for i = σ[u] + 1 to σ[w], do
            [c̄_i, d̄_i] ← [x_i^bR, x_i^aR];
        (x^LR_{σ[v−1]+1}, ..., x^LR_{σ[w]}) ← RAP_{v,w}(L, R, c̄, d̄).
```

Two final discussions follow.

- First, observe the occurrence of Algorithm 1 (Adjust function introduced in Proof of Theorem 2) before setting the RAP bounds. This $\mathbb{O}(n)$ time function can only occur when the functions f_i are not strictly convex; in these cases, the solutions of the subproblems may not directly satisfy $\mathbf{x^{La}} \leq \mathbf{x^{Lb}}$ and $\mathbf{x^{bR}} \leq \mathbf{x^{aR}}$ because of possible ties between resource allocation choices. Alternatively, one could also use a *stable* RAP solver that guarantees that the solution variables increase monotonically with the resource bound.
- Second, note the occurrence of the L1 penalty function associated with the original variables' bounds c_i and d_i in $\bar{f}_i(x_i)$, whereas $\bar{c}_i$ and $\bar{d}_i$ are maintained as hard constraints. Indeed, some subproblems (e.g., RAP–NC$_{v,v+1}(b_v, a_{v+1})$

when $b_v \geq a_{v+1}$ and $\mathbf{c} = \mathbf{0}$) may not have a solution respecting the bounds c_i and d_i. However, the $\bar{c}_i$ and $\bar{d}_i$ constraints can always be fulfilled; otherwise, the original problem would be infeasible, and their validity is essential to guarantee the correctness of the algorithm.

Nevertheless, because efficient RAP algorithms exist for some specific forms of the objective function (e.g., quadratic) (Brucker 1984, Ibaraki and Katoh 1988), we wish to avoid explicit penalty terms in the objective. Therefore, we note that an optimal solution $\mathbf{x}^*$ of $\text{RAP}_{v,w}(L, R, \bar{\mathbf{c}}, \bar{\mathbf{d}})$ can be obtained as follows:

$$\mathbf{x}^* = \begin{cases} \mathbf{c}' + \frac{(R-L)-\sum_{i=\sigma[v-1]+1}^{\sigma[w]} c'_i}{\sum_{i=\sigma[v-1]+1}^{\sigma[w]} (\bar{c}_i - c'_i)} (\bar{\mathbf{c}} - \mathbf{c}') & \text{if } \sum_{i=\sigma[v-1]+1}^{\sigma[w]} c'_i > R - L \\ \mathbf{d}' + \frac{(R-L)-\sum_{i=\sigma[v-1]+1}^{\sigma[w]} d'_i}{\sum_{i=\sigma[v-1]+1}^{\sigma[w]} (\bar{d}_i - d'_i)} (\bar{\mathbf{d}} - \mathbf{d}') & \text{if } \sum_{i=\sigma[v-1]+1}^{\sigma[w]} d'_i < R - L \\ \mathbf{x} & \text{otherwise,} \end{cases} \tag{59}$$

where $c'_i = \max\{\bar{c}_i, \min\{c_i, \bar{d}_i\}\}$, $d'_i = \min\{\bar{d}_i, \max\{d_i, \bar{c}_i\}\}$, and $\mathbf{x}$ is the solution of the same RAP with the hard constraints of Equation (60):

$$c'_i \leq x_i \leq d'_i \qquad i \in \{\sigma[v-1]+1, \ldots, \sigma[w]\}. \tag{60}$$

Thus, the penalty functions for c_i and d_i are taken into account by a $\mathbb{O}(n)$ test during each RAP resolution, and they never appear in the objective. Experimentally, we observe that the subproblems that fall in the first two cases of Equation (59) are solved notably faster, because they do not even require finding the minimum of a convex function.

3.2. Integer Optimization and Proximity

The previous section has considered continuous decision variables and proven the validity of the algorithm when all of the subproblems are *solved to optimality*. Still, this proof is of limited practical utility for bit complexity computational models, because the solutions of separable convex problems can involve irrational numbers (e.g., $\min f(x) = x^3 - 6x, x \geq 0$), which have no finite binary representation. Therefore, assuming that a subproblem is solved to optimality without any assumption on the shape of the functions is impracticable.

For this reason, most articles that present computational complexity results for convex resource allocation and network flow problems rely on the notion of ϵ-approximate solutions located in the proximity of a truly optimal but not necessarily representable solution. In a decomposition algorithm, such as MDA, proving that the method produces an ϵ-approximate solution for a given ϵ would require us to control the precision of the algorithm at each level of the recursion, which could be cumbersome. Therefore, we adopt another approach that is typically used in scaling algorithms (Hochbaum 1994, Moriguchi et al. 2011), which consists of proving the validity of the algorithm for integer variables and using a proximity theorem between the integer and continuous solutions. By solving an integer problem scaled by an appropriate factor and translating back the integer solution into a continuous solution, any desired ϵ precision can be achieved.

We define the functions $\bar{f}_i^{\text{PL}}(x) = \bar{f}_i(\lfloor x \rfloor) + (x - \lfloor x \rfloor) \times (\bar{f}_i(\lceil x \rceil) - \bar{f}_i(\lfloor x \rfloor))$, which correspond to an inner linearization of the objective using as base the set of integer values. We call the linearized problem $\text{RAP–NC}_{v,w}^{\text{PL}}(L, R)$; it aims to find the minimum of $\bar{f}^{\text{PL}}(\mathbf{x}) = \sum_{i=\sigma[v-1]+1}^{\sigma[w]} \bar{f}_i^{\text{PL}}(x_i)$ subject to Equations (40)–(42). Because $\bar{f}_i$ and $\bar{f}_i^{\text{PL}}$ coincide on the integer domain, the integers $\text{RAP–NC}_{v,w}(L, R)$ and $\text{RAP–NC}_{v,w}^{\text{PL}}(L, R)$ have the same set of optimal solutions. Beyond this, there is a close relationship between the solutions of the integer $\text{RAP–NC}_{v,w}^{\text{PL}}(L, R)$ and those of its continuous counterpart as formulated in Theorem 4, allowing us to prove the validity of Algorithm 2 for integer variables (Theorem 5).

Theorem 4 (Reformulation). *Any optimal solution* $\mathbf{x}^*$ *of the integer* $\text{RAP–NC}_{v,w}^{\text{PL}}(L, R)$ *is also an optimal solution of the continuous* $\text{RAP–NC}_{v,w}^{\text{PL}}(L, R)$.

Proof. By contradiction, suppose that $\mathbf{x}^*$ is not an optimal solution of the continuous $\text{RAP–NC}_{v,w}^{\text{PL}}(L, R)$. Hence, there exists $\mathbf{x}$ such that $\bar{f}^{\text{PL}}(\mathbf{x}) < \bar{f}^{\text{PL}}(\mathbf{x}^*)$, and the set $\{i \mid x_i - \lfloor x_i \rfloor > 0\}$ contains at least two elements, because $\sum_{i=\sigma[v-1]+1}^{\sigma[w]} x_i = R - L \in \mathbb{Z}$. Let s and t be the first and second indices in this set, respectively. We know that the functions f_s^{PL} and f_t^{PL} are linear in $[\lfloor x_s \rfloor, \lceil x_s \rceil]$ and $[\lfloor x_t \rfloor, \lceil x_t \rceil]$, respectively, with slopes Φ_s and Φ_t. Observe that the solution,

$$\mathbf{x}' = \begin{cases} \mathbf{x} + \min\{\lceil x_s \rceil - x_s, x_t - \lfloor x_t \rfloor\}(\mathbf{e}^s - \mathbf{e}^t) & \text{if } \Phi_s \leq \Phi_t, \\ \mathbf{x} + \min\{\lceil x_t \rceil - x_t, x_s - \lfloor x_s \rfloor\}(\mathbf{e}^t - \mathbf{e}^s) & \text{otherwise,} \end{cases} \tag{61}$$

is feasible such that $\bar{f}^{\mathrm{PL}}(\mathbf{x}') \leq \bar{f}^{\mathrm{PL}}(\mathbf{x})$. Also, note that the number of noninteger values of $\mathbf{x}'$ has been strictly decreased (by one or two). Repeating this process, we obtain an integer solution $\mathbf{x}^{**}$ such that $\bar{f}^{\mathrm{PL}}(\mathbf{x}^{**}) \leq \bar{f}^{\mathrm{PL}}(\mathbf{x}) < \bar{f}^{\mathrm{PL}}(\mathbf{x}^*)$. This contradicts the original assumption that $\mathbf{x}^*$ is an optimal solution of the integer RAP–NC$^{\mathrm{PL}}_{v,w}(L,R)$. □

Theorem 5 (Integer Variables). *Theorems 1–3 and Algorithm 2 remain valid for RAP–NCs with integer variables.*

Proof. The mathematical arguments used in these proofs are independent of the continuous or integer nature of the variables. Moreover, the solution transformation of Algorithm 1 preserves the integrality of the variables. The only element that requires continuous variables is the use of the (necessary) KKT conditions in Equations (44)–(47). However, as we have shown in Theorem 4, an optimal solution of the RAP–NC$^{\mathrm{PL}}_{v,w}(L,R)$ with integer variables is also an optimal solution of the continuous RAP–NC$^{\mathrm{PL}}_{v,w}(L,R)$. Thus, the KKT conditions with functions f_i^{PL} are necessary, hence completing the proof. □

Finally, we exploit a proximity result for the solutions of the continuous and integer RAP–NC$^{\mathrm{PL}}_{v,w}(L,R)$.

Theorem 6 (Proximity). *For any integer optimal solution* $\mathbf{x}^*$ *of* RAP–NC *with* $n \geq 2$ *variables, there is a continuous optimal solution* $\mathbf{x}$ *such that*

$$|x_i - x_i^*| < n-1 \text{ for } i \in \{1,\ldots,n\}. \tag{62}$$

This theorem allows us to search for an ϵ-approximate solution of the continuous problem by defining an integer RAP–NC in which all parameters (a_i, b_i, c_i, and d_i) have been scaled by a factor $\lceil n/\epsilon \rceil$, solving this problem, and transforming back the solution. It constitutes a special case of theorem 1.3 from Moriguchi et al. (2011), because the RAP–NC can be shown to be a special case of the resource allocation problem under submodular constraints. Moreover, without even relying on submodular optimization arguments, this result can also be obtained directly via first-order (KKT) optimality conditions. This alternative proof shares many similarities with that of Theorem 3, and it is made available in the appendix for the interested reader.

3.3. Computational Complexity

Convex Objective. Each call to the main algorithm MDA(v,w) involves a recursive call to MDA(v,u) and MDA$(u+1,w)$ with $u = \lfloor \frac{v+w}{2} \rfloor$ as well as

- the solution of RAP$_{v,w}(L,R,\bar{\mathbf{c}},\bar{\mathbf{d}})$ for $L \in \{a_{v-1}, b_{v-1}\}$ and $R \in \{a_w, b_w\}$;
- up to four calls to the ADJUST function;
- a linear number of operations to set the bounds $\bar{c}_i$ and $\bar{d}_i$.

The function ADJUST uses a number of elementary operations that grow linearly with the number of variables. Moreover, in the presence of integer variables, each RAP subproblem with n variables and bound $B = R - L$ can be solved in $\mathbb{O}(n \log \frac{B}{n})$ time using the algorithm of Frederickson and Johnson (1982) or Hochbaum (1994). As a consequence, the number of operations $\Phi(n,m,B)$ of MDA as a function of the number of variables n and constraints m is bounded as

$$\begin{aligned}\Phi(n,m,B) &\leq \sum_{i=1}^{h}\left(Kn + \sum_{j=1}^{2^{h-i}} 4K'\left(\sigma[2^i j] - \sigma[2^i(j-1)]\right)\log\left(\frac{B}{\sigma[2^i j] - \sigma[2^i(j-1)]}\right)\right)\\ &\leq Knh + 4K'nh\log B,\end{aligned}$$

where K and K' are constants and $h = 1 + \lceil \log_2 m \rceil$. Thus, $\Phi(n,m,B) \in \mathbb{O}(n \log m \log B)$ in the integer case. For the continuous case, after scaling all problem parameters by $\lceil n/\epsilon \rceil$, the complexity of the algorithm for the search for an ϵ−approximate solution becomes $\mathbb{O}(n \log m \log \frac{nB}{\epsilon})$.

Quadratic and Linear Objectives. More efficient RAP solution methods are known for specific forms of objective functions. The quadratic RAP with continuous variables, in particular, can be solved in $\mathbb{O}(n)$ time (Brucker 1984). For the integer case, reviewed in Katoh et al. (2013), an $\mathbb{O}(n)$ algorithm can be derived from section 4.6 of Ibaraki and Katoh (1988). Finally, in the linear case, each RAP subproblem can be solved in $\mathbb{O}(n)$ time as a weighted median problem (see, e.g., Korte and Vygen 2012). All of these cases lead to $\mathbb{O}(n \log m)$ algorithms for the corresponding RAP–NC. Note that no transformation or proximity theorem is needed for the continuous quadratic RAP, because the solutions of quadratic problems are representable.

4. Computational Experiments

We perform computational experiments to evaluate the performance of the proposed algorithm in the presence of a linear objective and for two convex objective functions arising in project crashing and speed optimization applications. For linear problems, we compare with the network flow algorithm of Ahuja and Hochbaum (2008), which achieved the previous best-known complexity of $\mathbb{O}(n \log n)$ for the problem; this complexity is slightly improved to $\mathbb{O}(n \log m)$ by the proposed MDA. For general convex objectives, no dedicated algorithm is available, and we compare with the interior point-based algorithm of MOSEK v7.1 for separable convex optimization. We finally report experimental analyses to evaluate the potential of this solver within a projected gradient method for the SVOREX problem (Section 2) for ordinal regression. The algorithms are implemented in C++ and executed on a single core of a Xeon 3.07 GHz CPU. For accurate time measurements, any algorithm with a CPU time smaller than one second was executed multiple times in a loop (up to a total time of 10 seconds) to determine the average time of a run.

We generated benchmark instances with a number of variables $n \in \{10, 20, 50, 100, 200, \ldots, 10^6\}$. Overall, 10 random benchmark instances were produced for each problem size, leading to a total of 16×10 instances with the same number of nested constraints as decision variables ($n = m$). For fine-grained complexity analyses in the case of the linear objective, we also removed random nested constraints to produce an additional set of 13×10 instances with $m = 100$ constraints and $n \in \{100, 200, 500, \ldots, 10^6\}$. For each instance, we generated the parameters c_i and d_i for $i \in \{1, \ldots, n\}$ from uniform distributions in the range $[0.1, 0.5]$ and $[0.5, 0.9]$, respectively. Then, we defined two sequences of values v_i and w_i such that $v_0 = w_0 = 0$, $v_i = v_{i-1} + X_i^v$, and $w_i = w_{i-1} + X_i^w$ for $i \in \{1, \ldots, n\}$, where X_i^v and X_i^w are random variables drawn from a uniform distribution in the range $[c_i, d_i]$. Finally, we set $a_i = \min\{v_i, w_i\}$ and $b_i = \max\{v_i, w_i\}$ for all i. We also selected a random parameter p_i in $[0, 1]$ to characterize the objective function. We conducted the experiments with four classes of objectives: a linear objective $\sum_{i=1}^{n} p_i x_i$ and three convex objectives defined as

$$\text{[F]} \qquad f_i(x) = \frac{x^4}{4} + p_i x, \tag{63}$$

$$\text{[Crash]} \qquad f_i(x) = K_i + \frac{p_i}{x}, \tag{64}$$

$$\text{and [Fuel]} \qquad f_i(x) = p_i \times c_i \times \left(\frac{c_i}{x}\right)^3, \tag{65}$$

where the last two objectives are representative of applications in project crashing (Foldes and Soumis 1993) and ship speed optimization (Ronen 1982), respectively.

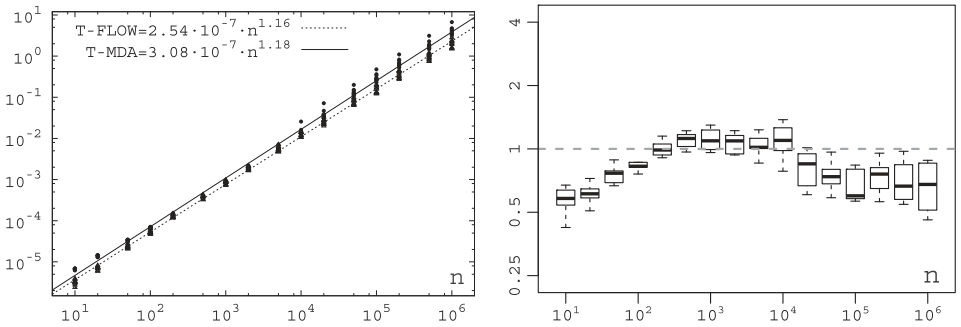
4.1. Linear Objective

We start the experimental analyses with the linear RAP–NC. We will refer to the network-flow-based approach of Ahuja and Hochbaum (2008) as "FLOW" in the text and tables. This method was precisely described in the original article, but no computational experiments or practical implementation were reported; therefore, we had to implement it. The authors suggest the use of a red-black tree to locate the minimum cost paths and a dynamic tree (Tarjan 1997, Tarjan and Werneck 2009) to manage the capacity constraints. This advanced data structure requires significant implementation effort and can result in high CPU time constants. We thus adopted a simpler structure, a segment tree (Bentley 1977) with lazy propagation, which allows for evaluating and updating these capacities with the same complexity of $\mathbb{O}(\log n)$ per operation (and possibly, a higher speed in practice). The proposed MDA was implemented as in Algorithm 2, solving each linear RAP subproblem in $\mathbb{O}(n)$ time as a variant of a weighted median problem (Korte and Vygen 2012).

We executed both algorithms on each instance. The results for the instances with as many nested constraints as decision variables ($n = m$) are reported in Figure 1. To evaluate the growth of the computational effort of the algorithms as a function of problem size, we fitted the computational time as a power law $f(n) = \alpha \times n^{\beta}$ of the number of variables n via a least squares regression of an affine function on the log-log graph (Figure 1, left panel). We also display as boxplots the ratio of the computational time of MDA and FLOW (Figure 1, right panel). The same conventions are used to display the results of the experimental analyses with a fixed number of constraints ($m = 100$) and increasing number of variables n in Figure 2. Finally, the detailed average computational times for each group of 10 instances are reported in Table 1.

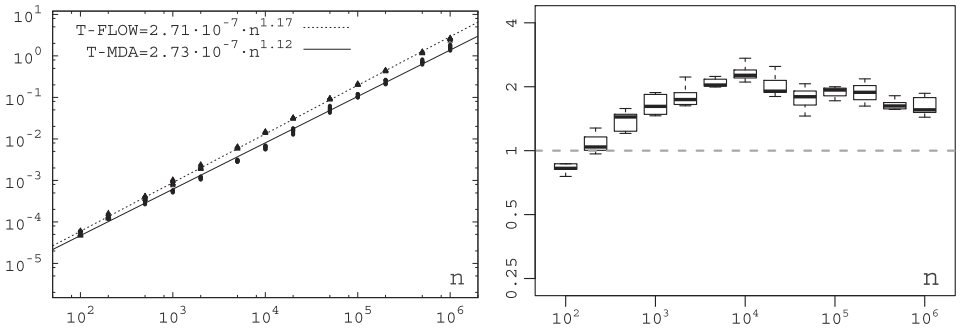
From these experiments, we observe that the computational times of the two methods are very similar in terms of magnitude and growth rate. When $m = n$, the algorithms have the same theoretical complexity of $\mathbb{O}(n \log n)$ as confirmed by the power law regression, with an observed growth that is close to linear (in $n^{1.16}$ and $n^{1.18}$).

Figure 1. Linear Objective: Varying $n \in \{10, \ldots, 10^6\}$ and $m = n$



Notes. (Left panel) CPU time of both methods as n and m grow. (Right panel) Boxplots of the ratio $T_{\text{FLOW}}/T_{\text{MDA}}$.

Figure 2. Linear Objective: Varying $n \in \{10, \ldots, 10^6\}$ and fixed $m = 100$



Notes. (Left panel) CPU time of both methods as n grows. (Right panel) Boxplots of the ratio $T_{\text{FLOW}}/T_{\text{MDA}}$.

The FLOW algorithm is on average from 1.1 to 1.4 times faster than MDA for $n \in [10, 100] \cup [10^4, 10^6]$ for instances with the same number of variables and constraints. However, MDA is on average two times faster than FLOW when m is fixed and n grows beyond 1,000. This is because of the difference in computational complexity: $\mathbb{O}(n \log m)$ for MDA instead of $\mathbb{O}(n \log n)$. MDA and FLOW solve the largest instances, with up to $n = m = 10^6$ constraints and variables, in three and two seconds on average, respectively.

Overall, the two algorithms have similar performance for linear objectives, and the CPU differences are small. Because these algorithms are based on drastically different principles, they lead the way to different methodological

Table 1. Detailed Average CPU Times—Experiments with a Linear Objective

Variable m		CPU time(s)		Fixed m		CPU time(s)	
n	m	FLOW	MDA	n	m	FLOW	MDA
10	10	2.75×10^{-6}	4.78×10^{-6}	100	100	5.09×10^{-5}	5.95×10^{-5}
20	20	6.26×10^{-6}	1.02×10^{-5}	200	100	1.36×10^{-4}	1.26×10^{-4}
50	50	2.15×10^{-5}	2.85×10^{-5}	500	100	3.94×10^{-4}	2.86×10^{-4}
100	100	5.06×10^{-5}	5.89×10^{-5}	1,000	100	9.07×10^{-4}	5.52×10^{-4}
200	200	1.26×10^{-4}	1.26×10^{-4}	2,000	100	2.07×10^{-3}	1.14×10^{-3}
500	500	3.72×10^{-4}	3.36×10^{-4}	5,000	100	6.16×10^{-3}	2.96×10^{-3}
1,000	1,000	8.43×10^{-4}	7.57×10^{-4}	10,000	100	1.44×10^{-2}	6.26×10^{-2}
2,000	2,000	1.87×10^{-3}	1.74×10^{-3}	20,000	100	3.17×10^{-2}	1.57×10^{-2}
5,000	5,000	5.43×10^{-3}	5.20×10^{-3}	50,000	100	9.27×10^{-1}	5.26×10^{-1}
10,000	10,000	1.23×10^{-2}	1.12×10^{-2}	100,000	100	2.04×10^{-1}	1.08×10^{-1}
20,000	20,000	2.62×10^{-2}	3.21×10^{-2}	200,000	100	4.41×10^{-1}	2.36×10^{-1}
50,000	50,000	7.94×10^{-2}	1.05×10^{-1}	500,000	100	1.20	7.19×10^{-1}
100,000	100,000	1.52×10^{-1}	2.26×10^{-1}	1,000,000	100	2.56	1.60
20,0000	200,000	3.67×10^{-1}	4.86×10^{-1}				
500,000	500,000	9.68×10^{-1}	1.37				
1,000,000	1,000,000	1.99	2.98				

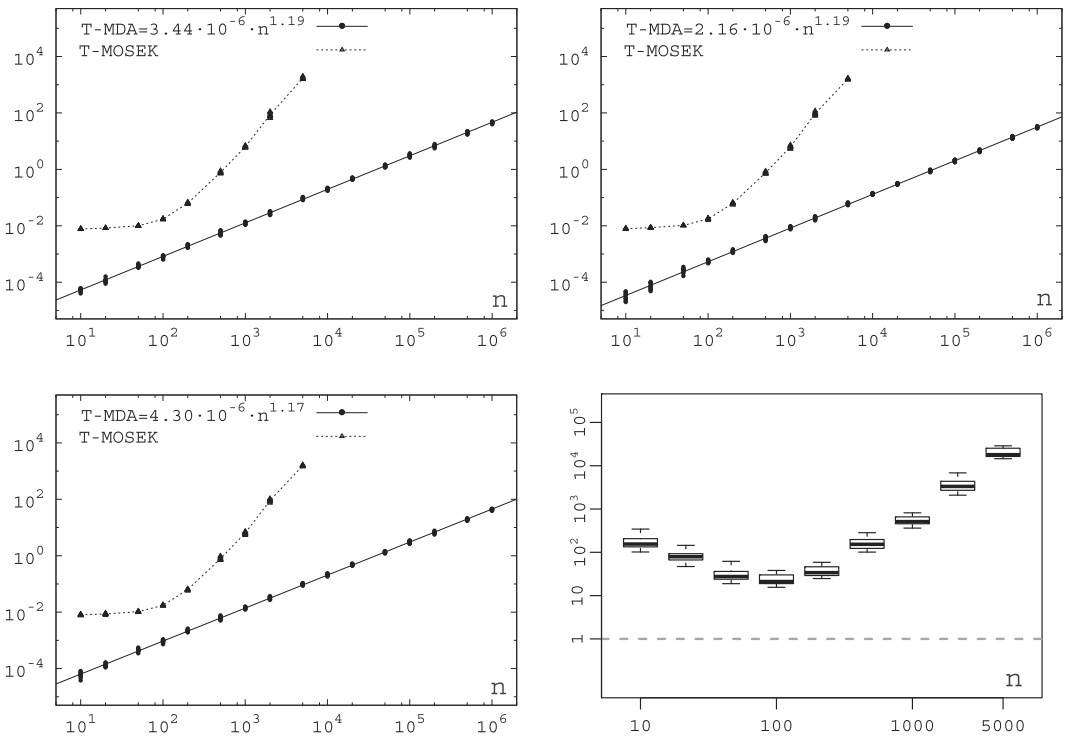
extensions. The computational complexity of FLOW is tied to its efficient use of a dynamic tree data structure, whereas the complexity of the MDA stems from its "monotonic" divide-and-conquer strategy. Because of this structure, MDA should be a good choice for reoptimization after a change of a few parameters as well as for the iterative solution of multiple RAP–NC: for example, for speed optimization within an algorithm enumerating a large number of similar visit sequences, because it can reuse the solutions of smaller subproblems (see, e.g., Norstad et al. 2011; Vidal et al. 2014, 2015).

4.2. Separable Convex Objectives

In contrast with the case of linear objective functions, no specialized algorithm has been designed for the RAP–NC with separable convex costs to date. To illustrate the possible gain achieved by the use of a dedicated algorithm rather than a general purpose solver, we do a simple comparison of the CPU time of MDA with that of the MOSEK v7.1 solver on two sets of instances with objective functions derived from project crashing and speed optimization applications. MOSEK is based on an interior point method, and it is a good representative of the current generation of separable convex optimization solvers. We set a time limit of one hour. To simplify the execution of these experiments, we use a binary search over the single dual variable to solve each continuous RAP subproblem (Patriksson 2008) within a precision of 10^{-9}. Because these approximations may stack up $\log m$ times in the recursion, we obtain an overall good accuracy but do not guarantee ϵ proximity in these tests.

The results are reported in Figure 3 and Table 2. In Figure 3, the power law regressions are presented only for MDA, because MOSEK does not exhibit polynomial behavior, likely because of the computational effort related to the initialization of the solver for small problems. In contrast, the computational time of MDA grows steadily in $\mathcal{O}(n^{1.19})$ at most. This observation is consistent with the theoretical $\mathcal{O}(n \log m)$ complexity of the method. Within one hour, MOSEK solved all of the instances up to $n = 5,000$ decision variables. In contrast, MDA solved all of the available instances with up to one million variables. For the largest benchmark instances, the CPU time of the method did not exceed 50 seconds. As illustrated in Figure 3, lower right panel, the ratio of the CPU time of MOSEK and MDA ranges between 16 and 28,000. For all of the instances, significant CPU time is saved when using the monotonic divide-and-conquer algorithm instead of a general purpose solver.

Figure 3. Convex Objectives



Notes. CPU time of both methods as n grows and $m = n$ for the objectives (upper left panel) [F], (upper right panel) [Crash], and (lower left panel) [Fuel]. (Lower right panel) Boxplots of the ratio $T_{\text{MOSEK}}/T_{\text{MDA}}$.

Table 2. Detailed Average CPU Time—Experiments with a Separable Convex Objective

		CPU time(s)—MDA			CPU time(s)—MOSEK		
n	m	[F]	[Crash]	[Fuel]	[F]	[Crash]	[Fuel]
10	10	5.28×10^{-5}	3.27×10^{-5}	6.11×10^{-5}	7.69×10^{-3}	7.83×10^{-3}	8.06×10^{-3}
20	20	1.14×10^{-4}	7.32×10^{-5}	1.33×10^{-4}	8.27×10^{-3}	8.60×10^{-3}	8.64×10^{-3}
50	50	3.80×10^{-4}	2.63×10^{-4}	4.45×10^{-4}	9.95×10^{-3}	1.03×10^{-2}	1.04×10^{-2}
100	100	8.04×10^{-4}	5.39×10^{-4}	9.30×10^{-4}	1.73×10^{-2}	1.75×10^{-2}	1.74×10^{-2}
200	200	1.93×10^{-3}	1.23×10^{-3}	2.16×10^{-3}	6.31×10^{-2}	6.22×10^{-2}	6.30×10^{-2}
500	500	1.93×10^{-3}	3.55×10^{-3}	6.21×10^{-3}	7.79×10^{-1}	7.56×10^{-1}	7.86×10^{-1}
1,000	1,000	5.45×10^{-3}	8.61×10^{-3}	1.43×10^{-2}	6.31	6.29	6.37
2,000	2,000	1.27×10^{-2}	1.87×10^{-2}	3.19×10^{-2}	8.57×1^{01}	9.38×1^{01}	9.05×1^{01}
5,000	5,000	2.88×10^{-2}	6.05×10^{-2}	9.86×10^{-2}	1.70×1^{03}	1.61×1^{03}	1.55×1^{03}
10,000	10,000	2.01×10^{-1}	1.34×10^{-1}	2.13×10^{-1}	—	—	—
20,000	20,000	4.69×10^{-1}	3.04×10^{-1}	4.82×10^{-1}	—	—	—
50,000	50,000	1.31	8.74×10^{-1}	1.33	—	—	—
100,000	100,000	3.12	2.02	3.07	—	—	—
200,000	200,000	6.68	4.58	6.61	—	—	—
500,000	500,000	1.98×1^{01}	1.35×1^{01}	1.91×1^{01}	—	—	—
1,000,000	1,000,000	4.54×1^{01}	3.10×1^{01}	4.30×1^{01}	—	—	—

4.3. Nonseparable Convex Objective—Support Vector Ordinal Regression

Our last experimental analysis is concerned with the SVOREX model presented in Section 2. It is a nonseparable convex optimization problem over a special case of the RAP–NC constraint polytope. The current state-of-the-art algorithm for this problem, proposed by Chu and Keerthi (2007), is based on a working set decomposition. Iteratively, a set of variables is selected to be optimized over, whereas the others remain fixed. This approach leads to a (nonseparable) restricted problem with fewer variables that can be solved to optimality. The authors rely on a *minimal working set* containing the two variables that most violate the KKT conditions (see Chu and Keerthi 2007, pp. 799–800, for all equations involved).

The advantage of a minimal working set comes from the availability of analytical solutions for the restricted problems. However, larger working sets can be beneficial to reduce the number of iterations until convergence (see, e.g., Joachims 1999). However, this would require an efficient method for the resolution of the reduced problems. This is how the proposed RAP–NC solver can provide a meaningful option along this direction. To evaluate such a proof of concept, we conduct a simple experiment that consists of generating larger working sets within the approach of Chu and Keerthi (2007) and solving the resulting reduced problems with the help of the RAP–NC algorithm. Because these reduced problems are nonseparable convex, the RAP–NC algorithm is being used for the projection steps within a projected gradient descent procedure. The overall solution approach is summarized in Algorithm 3, in which $\mathcal{W}$ is the working set, z is the objective function, and γ is the fixed step size of the gradient descent.

Algorithm 3 (Solving SVOREX via RAP–NC subproblems)

$\boldsymbol{\alpha} = \boldsymbol{\alpha}^* = \mathbf{0}$
while there exists samples that violate the KKT conditions, **do**
  select a working set $\mathcal{W}$ of maximum size n_{ws}
  for n_{grad} iterations, **do**
    // Take a step
    for $j \in \{1, \dots, r\}$ and $i \in \{1, \dots, n^j\}$, **do**

$$\hat{\alpha}_i^j = \begin{cases} \alpha_i^j + \gamma \dfrac{\partial z}{\partial \alpha_i^j} & \text{if } (i,j) \in \mathcal{W} \\ \alpha_i^j & \text{otherwise} \end{cases}; \quad \hat{\alpha}_i^{*j} = \begin{cases} \alpha_i^{*j} + \gamma \dfrac{\partial z}{\partial \alpha_i^{*j}} & \text{if } (i,j) \in \mathcal{W} \\ \alpha_i^{*j} & \text{otherwise} \end{cases}$$

    // Solve the projection subproblem as an RAP–NC

$$(\boldsymbol{\alpha}, \boldsymbol{\alpha}^*) \leftarrow \begin{cases} \min\limits_{\boldsymbol{\alpha}, \boldsymbol{\alpha}^*} & \sum\limits_{(i,j) \in \mathcal{W}} \left((\alpha_i^j - \hat{\alpha}_i^j)^2 + (\alpha_i^{*j} - \hat{\alpha}_i^{*j})^2 \right) \\ \text{s.t.} & \text{Equations (30)–(33)} \\ & \alpha_i^j = \hat{\alpha}_i^j \text{ and } \alpha_i^{*j} = \hat{\alpha}_i^{*j} \text{ if } (i,j) \notin \mathcal{W}. \end{cases}$$

To obtain a larger working set, we repeatedly select the most violated sample pair until either reaching the desired size or not finding any remaining violation. In our experiments, we consider working sets of size $n_{ws} \in \{2, 4, 6, 10\}$, a step size of $\gamma = 0.2$, and $n_{GRAD} = 20$ iterations for the projected gradient descent. We use the eight datasets as Chu and Keerthi (2007) with the same Gaussian kernel, penalty parameter, and guidelines for data preparation (normalizing the input vectors to zero mean and unit variance and using equal frequency binning to discretize the target values into five ordinal scales).

Table 3 gives the results of these experiments. The columns report, in turn, the name of the dataset, its number of samples N, the dimension D of its feature space, and characteristics of the optimal solutions: the numbers of variables set to zero (correct classification), C (misclassified), and intermediate values (support vectors). For each working set size n_{ws}, the total number of working set selections I_{ws} done by the algorithm is also presented as well as the CPU time in seconds. The fastest algorithm version is bold for each dataset.

As measured in these experiments, the CPU time of the algorithms ranges between 0.018 seconds for the smallest datasets (with 50 samples and 27 dimensions) and 574.28 seconds for the largest case (6,000 samples and 16 dimensions). The size of the working set has a significant impact on the number of iterations of the method and its CPU time.

In all cases, the number of iterations decreases significantly when the size of the working set grows. In terms of CPU time, the fastest results are achieved with a working set of either size two or size six (with three datasets in each case). We observe that the three datasets for which a larger working set contributed to CPU time reductions are those with higher-dimension feature spaces (dimension 21–32). For these datasets, using a larger working set helped reduce the CPU time by a factor of 1.1–1.9 compared with using a two-samples working set. In comparison,

Table 3. SVOREX Resolution—Impact of the Working Set Size and Solution Features

			Solution variables s.t., %					
Dataset	N	D	$\alpha = 0$	$\alpha = C$	$\alpha \in]0, C[$	n_{ws}	I_{ws}	T(s)
Abalone	1,000	8	39	32	29	**2**	**118,233**	**13.46**
						4	96,673	21.51
						6	78,433	26.34
						10	60,605	35.46
Bank	3,000	32	25	0	75	2	139,468	68.41
						4	52,073	63.02
						6	**31,452**	**45.22**
						10	21,310	47.66
Boston	300	13	41	0	59	2	7,207	0.43
						4	**3,697**	**0.40**
						6	2,840	0.46
						10	2,076	0.54
California	5,000	8	51	43	6	**2**	**250,720**	**124.46**
						4	189,289	185.79
						6	166,879	245.08
						10	146,170	360.52
Census	6,000	16	38	4	59	**2**	**349,894**	**242.11**
						4	206,951	301.74
						6	180,608	393.28
						10	155,731	574.28
Computer	4,000	21	64	32	4	2	290,207	168.94
						4	140,270	161.45
						6	**98,948**	**153.56**
						10	68,616	193.10
Machine CPU	150	6	49	9	41	2	28,856	1.24
						4	**11,534**	**0.86**
						6	8,144	0.91
						10	6,363	1.24
Pyrimidines	50	27	21	0	79	2	935	0.035
						4	367	0.021
						6	**218**	**0.018**
						10	144	0.023

Notes. The fastest algorithm version is indicated in bold for each dataset. T(s), time in seconds.

Joachims (1999) reported a speedup of 1.5–2.0 when using 10-samples working sets for SVM, which can be viewed as a special case of SVOREX with two classes.

To achieve a gain in CPU time, the number of iterations should decrease more than linearly as a function of the working set size. This is because of the effort spent updating the gradient that is necessary for the verification of the KKT conditions and the working set selection, which grows linearly with the product $I_{\text{ws}} \times N \times n_{\text{ws}}$ (using efficient incremental updates) and remains a major bottleneck for SVOREX and SVM algorithms (see, e.g., the discussions in Joachims 1999 and Chang and Lin 2011). Usually, datasets with a feature space of high dimension exhibit a fast decrease in the number of iterations as a function of the working set size, because their solutions include a larger proportion of variables taking values in $(0, C)$ (support vectors), values that are more quickly reached via simultaneous optimizations of several variables. As such, larger working sets are likely to be more useful in feature spaces of higher dimension.

Moreover, future research avenues concern possible improvements of the algorithm (e.g., using shrinking or a double-loop scheme) (Keerthi et al. 2001) and adaptive choices of working set size based on analyses of the structure of the data set and solutions as well as more advanced selection rules (e.g., based on Zoutendijk's descent direction (Joachims 1999) or second-order information (Fan et al. 2005). These options for improvement are now possible because of the availability of a fast algorithm for the resolution of the restricted problems.

5. Concluding Remarks

In this article, we have highlighted the importance of the RAP–NC, which is a problem connected with a wide range of applications in production and transportation optimization, portfolio management, sampling optimization, telecommunications, and machine learning. To solve this problem, we proposed a decomposition algorithm based on monotonicity principles coupled with divide and conquer, leading to new complexity breakthroughs for a variety of objectives (linear, quadratic, and convex) with continuous or integer variables and to the first known strongly polynomial algorithm for the quadratic integer RAP–NC. In terms of practical performance, the algorithm matches the best-dedicated (flow-based) algorithm for the linear case, outperforms general purpose solvers by several orders of magnitude in the convex case, and opens interesting perspectives of algorithmic improvements for the SVOREX problem in machine learning.

The algorithm can be seen as a generalization of the method of Vidal et al. (2016), with some fundamental differences related to the number and the nature of the subproblems. It is not based on classical greedy steps and scaling or on flow propagation techniques, which are often exploited for this problem family. It is an important research question to see how far this decomposition technique can be generalized. In particular, the approach can very likely be extended to the resource allocation problem with a TREE of lower and upper constraints (Hochbaum 1994). Other optimization problems related to Program Evaluation and Review Technique (PERT) may exhibit monotonicity properties as a function of time constraints or budget bounds, and we should investigate how to decompose efficiently their variables and constraints while maintaining a low computational complexity. Similarly, extended formulations involving the intersection of two or more RAP–NC types of constraint polytopes deserve a closer look. These are all open important research directions that can be explored in the near future.

Appendix. Proof of Theorem 6 Based on KKT Conditions

Proof. The proof shares many similarities with that of Theorem 2. It exploits the fact that an integer solution $\mathbf{x}^*$ of the RAP–NC$^{\text{PL}}$ is also an optimal solution of the continuous problem (Theorem 4) and thus, satisfies the KKT conditions of Equations (44)–(47) based on the functions f_i^{PL}. We first state two lemmas that will be used later to link the values of the subderivatives of f_i and f_i^{PL}.

Lemma 1. *Consider $y \in \mathbb{R}$ and $x \in \mathbb{Z}$ such that $y + 1 \le x$ and a convex function f. If $\phi_y \in \partial f(y)$ and $\phi_x \in \partial f^{\text{PL}}(x)$, then $\phi_y \le \phi_x$.*

Proof of Lemma 1. By definition, $f(x) - f(y) \ge \phi_y(x - y)$ and $f^{\text{PL}}(x-1) - f^{\text{PL}}(x) \ge \phi_x(x - 1 - x) = -\phi_x$. Moreover, $y \le x - 1 \le x$, f is convex, and f coincides with f^{PL} at x and $x - 1$; therefore,

$$\phi_y \le \frac{f(x) - f(y)}{x - y} \le \frac{f(x) - f(x-1)}{x - (x-1)} = f^{\text{PL}}(x) - f^{\text{PL}}(x-1) \le \phi_x. \quad \square$$

Lemma 2. *Consider $y \in \mathbb{Z}$ and $x \in \mathbb{R}$ such that $y + 1 \le x$ and a convex function f. If $\phi_y \in \partial f^{\text{PL}}(y)$ and $\phi_x \in \partial f(x)$, then $\phi_y \le \phi_x$.*

Proof of Lemma 2. By definition, $f^{\mathrm{PL}}(y+1) - f^{\mathrm{PL}}(y) \geq \phi_y$ and $f(y) - f(x) \geq \phi_x(y-x)$. Moreover, $y \leq y+1 \leq x$, f is convex, and f coincides with f^{PL} at y and $y+1$; therefore,

$$\phi_y \leq f^{\mathrm{PL}}(y+1) - f^{\mathrm{PL}}(y) = \frac{f(y+1) - f(y)}{(y+1)-y} \leq \frac{f^{\mathrm{PL}}(x) - f^{\mathrm{PL}}(y)}{x-y} \leq \phi_x. \quad \square$$

The main proof follows. Let $\mathbf{x}$ be an optimal continuous solution of the RAP–NC. If Equation (62) is satisfied, then the proof is complete; otherwise, there exists $s \in \{1,\ldots,n\}$ such that $|x_s - x_s^*| \geq n-1$. We consider here the case where $x_s \geq x_s^* + n - 1$, with the other case being symmetric. Let r be the greatest index in $\{1,\ldots,s\}$ such that $\sum_{k=1}^{r-1} x_k \geq \sum_{k=1}^{r-1} x_k^*$, and let t be the smallest index in $\{s,\ldots,n\}$ such that $\sum_{k=1}^{t} x_k \leq \sum_{k=1}^{t} x_k^*$. By the definition of r and t, it follows that $\sum_{i=r}^{t} x_i^* \geq \sum_{i=r}^{t} x_i$, and thus, $\sum_{i\in\{r,\ldots,t\}-s} x_i^* \geq \sum_{i\in\{r,\ldots,t\}-s} x_i + n - 1$. Because $|\{r,\ldots,t\} - s| \leq n-1$, there exists $u \in \{r,\ldots,t\} - s$ such that $x_u^* \geq x_u + 1$.

Two cases can arise.

If $u < s$, for each j such that $\sigma[j] \in \{u,\ldots,s-1\}$, $\bar{a}_j \leq \sum_{k=1}^{\sigma[j]} x_k < \sum_{k=1}^{\sigma[j]} x_k^* \leq \bar{b}_j$, and thus, $\kappa_j^* = \lambda_j = 0$. As a consequence, $\Phi_i \geq \Phi_{i+1}$ and $\Phi_i^* \leq \Phi_{i+1}^*$ for $i \in \{u,\ldots,s-1\}$.

If $u > s$, for each j such that $\sigma[j] \in \{s,\ldots,u-1\}$, $\bar{a}_j \leq \sum_{k=1}^{\sigma[j]} x_k^* < \sum_{k=1}^{\sigma[j]} x_k \leq \bar{b}_j$, and thus, $\lambda_j^* = \kappa_j = 0$. As a consequence, $\Phi_i \leq \Phi_{i+1}$ and $\Phi_i^* \geq \Phi_{i+1}^*$ for $i \in \{s,\ldots,u-1\}$.

Moreover, $\{x_s^* + n - 1 \leq x_s, \Phi_s^* \in \partial f_s(x_s^*), \Phi_s \in \partial f_s^{\mathrm{PL}}(x_s)\} \Rightarrow \Phi_s^* \leq \Phi_s$ (Lemma 1), and $\{x_u + 1 \leq x_u^*, \Phi_u \in \partial f_u^{\mathrm{PL}}(x_u), \Phi_u^* \in \partial f_u(x_u^*)\} \Rightarrow \Phi_u \leq \Phi_u^*$ (Lemma 2). Combining all of the relations leads to $\Phi_s \leq \Phi_u \leq \Phi_u^* \leq \Phi_s^* \leq \Phi_s$, and thus, there exists $\Psi \in \mathfrak{R}$ such that $\Phi_i^* = \Phi_i = \Psi$ for $i \in \{u,\ldots,s\}$ if $u < s$ (or $i \in \{s,\ldots,u\}$ if $s < u$). As in Proof of Theorem 2, this implies that the functions f_s and f_u are affine with slope Ψ over $[x_s^{\mathrm{MIN}}, x_s^{\mathrm{MAX}}]$ and $[x_u^{\mathrm{MIN}}, x_u^{\mathrm{MAX}}]$, respectively, where $x_i^{\mathrm{MIN}} = \min\{x_i, x_i^*\}$. Observe that the new solution $\mathbf{x}' = \mathbf{x} - \mathbf{e}^s + \mathbf{e}^u$ is a feasible solution with the same cost as $\mathbf{x}$ (hence, optimal). Moreover, we note that $\sum_{i=1}^{n} \max\{|x_i' - x_i^*| - (n-1), 0\} \leq \sum_{i=1}^{n} \max\{|x_i - x_i^*| - (n-1), 0\} - 1$ and/or $\sum_{i=1}^{n} \mathbb{1}\{|x_i' - x_i^*| \leq (n-1)\} \leq \sum_{i=1}^{n} \mathbb{1}\{|x_i - x_i^*| \leq (n-1)\} - 1$, where $\mathbb{1}(p) = 1$ if and only if p is true. Repeating this process leads, in a finite number of steps, to a solution $\mathbf{x}''$ such that $|x_i'' - x_i^*| < n-1$ for $i \in \{1,\ldots,n\}$. $\square$

References

Ahuja R, Hochbaum D (2008) Technical note—Solving linear cost dynamic lot-sizing problems in $O(n \log n)$ time. *Oper. Res.* 56(1):255–261.

Anagnostopoulos K, Mamanis G (2010) A portfolio optimization model with three objectives and discrete variables. *Comput. Oper. Res.* 37(7): 1285–1297.

Bektas T, Laporte G (2011) The pollution-routing problem. *Transportation Res. Part B: Methodological* 45(8):1232–1250.

Bellman R, Glicksberg I, Gross O (1954) The theory of dynamic programming as applied to a smoothing problem. *J. Soc. Indust. Appl. Math.* 2(2): 82–88.

Bentley J (1977) Solutions to Klee's rectangle problems. Technical report, Carnegie-Mellon University, Pittsburgh.

Bertsekas D, Nedi A, Ozdaglar A (2003) *Nonlinear Programming* (Athena Scientific, Nashua, NH).

Bienstock D (1996) Computational study of a family of mixed-integer quadratic programming problems. *Math. Programming* 74(2):121–140.

Brucker P (1984) An O(n) algorithm for quadratic knapsack problems. *Oper. Res. Lett.* 3(3):163–166.

Chang CC, Lin CJ (2011) LIBSVM: A library for support vector machines. *ACM Trans. Intelligent Systems Tech.* 2(27):1–27.

Chang T, Meade N, Beasley J, Sharaiha Y (2000) Heuristics for cardinality constrained portfolio optimisation. *Comput. Oper. Res.* 27(13): 1271–1302.

Chu W, Keerthi S (2007) Support vector ordinal regression. *Neural Comput.* 19(3):792–815.

Cortes C, Vapnik V (1995) Support-vector networks. *Machine Learn.* 20(3):273–297.

Crama Y, Schyns M (2003) Simulated annealing for complex portfolio selection problems. *Eur. J. Oper. Res.* 150(3):546–571.

D'Amico A, Sanguinetti L, Palomar D (2014) Convex separable problems with linear constraints in signal processing and communications. *IEEE Trans. Signal Processing* 62(22):6045–6058.

Fan RE, Chen PH, Lin CJ (2005) Working set selection using second order information for training support vector machines. *J. Machine Learn. Res.* 6:1889–1918.

Federgruen A, Groenevelt H (1986) The greedy procedure for resource allocation problems: Necessary and sufficient conditions for optimality. *Oper. Res.* 34(6):909–918.

Foldes S, Soumis F (1993) PERT and crashing revisited: Mathematical generalizations. *Eur. J. Oper. Res.* 64(2):286–294.

Frederickson G, Johnson D (1982) The complexity of selection and ranking in $X + Y$ and matrices with sorted columns. *J. Comput. System Sci.* 24(2): 197–208.

Gerards M, Hurink J, Hölzenspies P (2016) A survey of offline algorithms for energy minimization under deadline constraints. *J. Scheduling* 19(1): 3–19.

Gutierrez P, Perez-Ortiz M, Sanchez-Monedero J, Fernandez-Navarro F, Hervas-Martinez C (2016) Ordinal regression methods: Survey and experimental study. *IEEE Trans. Knowledge Data Engrg.* 28(1):127–146.

Hartley HO (1965) Multiple purpose optimum allocation in stratified sampling. *Proc. 125th Annual Meeting Amer. Statist. Assoc., Philadelphia, Pennsylvania*, 258–261.

Hochbaum D (1994) Lower and upper bounds for the allocation problem and other nonlinear optimization problems. *Math. Oper. Res.* 19(2): 390–409.

Huang W, Wang Y (2009) An optimal speed control scheme supported by media servers for low-power multimedia applications. *Multimedia Systems* 15(2):113–124.

Huddleston H, Claypool P, Hocking R (1970) Optimal sample allocation to strata using convex programming. *J. Royal Statist. Soc. Ser. C* 19(3): 273–278.

Hvattum L, Norstad I, Fagerholt K, Laporte G (2013) Analysis of an exact algorithm for the vessel speed optimization problem. *Networks* 62(2): 132–135.

Ibaraki T, Katoh N (1988) *Resource Allocation Problems: Algorithmic Approaches* (MIT Press, Boston).
Joachims T (1999) Making large-scale SVM learning practical. Burges C, Schölkopf B, Smola A, eds. *Advances in Kernel Methods* (MIT Press, Cambridge, MA), 169–184.
Jobst N, Horniman M, Lucas C, Mitra G (2001) Computational aspects of alternative portfolio selection models in the presence of discrete asset choice constraints. *Quant. Finance* 1(5):489–501.
Katoh N, Shioura A, Ibaraki T (2013) Resource allocation problems. Pardalos P, Du DZ, Graham R, eds. *Handbook of Combinatorial Optimization* (Springer, New York), 2897–2988.
Keerthi S, Shevade S, Bhattacharyya C, Murthy K (2001) Improvements to Platt's SMO algorithm for SVM classifier design. *Neural Comput.* 13(3): 637–649.
Kolm P, Tütüncü R, Fabozzi F (2014) 60 Years of portfolio optimization: Practical challenges and current trends. *Eur. J. Oper. Res.* 234(2):356–371.
Korte B, Vygen J (2012) The knapsack problem. *Combinatorial Optimization: Theory and Algorithms* (Springer, Berlin), 459–470.
Kramer R, Maculan N, Subramanian A, Vidal T (2015a) A speed and departure time optimization algorithm for the pollution-routing problem. *Eur. J. Oper. Res.* 247(3):782–787.
Kramer R, Subramanian A, Vidal T, Cabral L (2015b) A matheuristic approach for the pollution-routing problem. *Eur. J. Oper. Res.* 243(2): 523–539.
Love S (1973) Bounded production and inventory models with piecewise concave costs. *Management Sci.* 20(3):313–318.
Mansini R, Ogryczak W, Speranza M (2014) Twenty years of linear programming based portfolio optimization. *Eur. J. Oper. Res.* 234(2):518–535.
Markowitz H (1952) Portfolio selection. *J. Finance* 7(1):77–91.
Moriguchi S, Shioura A, Tsuchimura N (2011) M-convex function minimization by continuous relaxation approach: Proximity theorem and algorithm. *SIAM J. Optim.* 21(3):633–668.
Neyman J (1934) On the two different aspects of the representative method: The method of stratified sampling and the method of purposive selection. *J. Royal Statist. Soc.* 97(4):558–625.
Norstad I, Fagerholt K, Laporte G (2011) Tramp ship routing and scheduling with speed optimization. *Transportation Res. Emerging Tech.* 19(5): 853–865.
Padakandla A, Sundaresan R (2009) Power minimization for CDMA under colored noise. *IEEE Trans. Commun.* 57(10):3103–3112.
Palomar D, Fonollosa J (2005) Practical algorithms for a family of waterfilling solutions. *IEEE Trans. Signal Processing* 53(2):686–695.
Patriksson M (2008) A survey on the continuous nonlinear resource allocation problem. *Eur. J. Oper. Res.* 185(1):1–46.
Patriksson M, Strömberg C (2015) Algorithms for the continuous nonlinear resource allocation problem – New implementations and numerical studies. *Eur. J. Oper. Res.* 243(3):703–722.
Platt J (1998) Fast training of support vector machines using sequential minimal optimization. Schölkopf B, Burges C, Smola A, eds. *Advances in Kernel Methods* (MIT Press, Cambridge, MA), 185–208.
Psaraftis H, Kontovas C (2013) Speed models for energy-efficient maritime transportation: A taxonomy and survey. *Transportation Res. Emerging Tech.* 26:331–351.
Psaraftis H, Kontovas C (2014) Ship speed optimization: Concepts, models and combined speed-routing scenarios. *Transportation Res. Emerging Tech.* 44:52–69.
Renegar J (1987) On the worst-case arithmetic complexity of approximating zeros of polynomials. *J. Complexity* 3(2):90–113.
Rockafellar R (1970) *Convex Analysis* (Princeton University Press, Princeton, NJ).
Ronen D (1982) The effect of oil price on the optimal speed of ships. *J. Oper. Res. Soc.* 33(11):1035–1040.
Sanathanan L (1971) On an allocation problem with multistage constraints. *Oper. Res.* 19(7):1647–1663.
Sedeño-Noda A, Gutiérrez J, Abdul-Jalbar B, Sicilia J (2004) An $O(T \log T)$ algorithm for the dynamic lot size problem with limited storage and linear costs. *Comput. Optim. Appl.* 28(3):311–323.
Srikantan K (1963) A problem in optimum allocation. *Oper. Res.* 11(2):265–273.
Tarjan R (1997) Dynamic trees as search trees via Euler tours, applied to the network simplex algorithm. *Math. Programming* 78(2):169–177.
Tarjan R, Werneck R (2009) Dynamic trees in practice. *J. Experiment. Algorithmics* 14:5–23.
Vidal T, Jaillet P, Maculan N (2016) A decomposition algorithm for nested resource allocation problems. *SIAM J. Optim.* 26(2):1322–1340.
Vidal T, Crainic T, Gendreau M, Prins C (2014) A unified solution framework for multi-attribute vehicle routing problems. *Eur. J. Oper. Res.* 234(3):658–673.
Vidal T, Crainic T, Gendreau M, Prins C (2015) Timing problems and algorithms: Time decisions for sequences of activities. *Networks* 65(2): 102–128.
Viswanath P, Anantharam V (2002) Optimal sequences for CDMA under colored noise: A Schur-saddle function property. *IEEE Trans. Inform. Theory* 48(6):1295–1318.
Wagner H, Whitin T (1958) Dynamic version of the economic lot size model. *Management Sci.* 5(1):89–96.