A DECOMPOSITION ALGORITHM FOR NESTED RESOURCE ALLOCATION PROBLEMS

THIBAUT VIDAL*, PATRICK JAILLET†, AND NELSON MACULAN‡

Abstract. We propose an exact polynomial algorithm for a resource allocation problem with convex costs and constraints on partial sums of resource consumptions, in the presence of either continuous or integer variables. No assumption of strict convexity or differentiability is needed. The method solves a hierarchy of resource allocation subproblems, whose solutions are used to convert constraints on sums of resources into new bounds for variables at higher levels. The resulting time complexity for the integer problem is $O(n \log m \log(B/n))$, and the complexity of obtaining an ϵ-approximate solution for the continuous case is $O(n \log m \log(B/\epsilon))$, n being the number of variables, m the number of ascending constraints (such that $m \leq n$), ϵ a desired precision, and B the total resource. This algorithm matches the best-known complexity when $m = n$, and improves it when $\log m = o(\log n)$. Extensive experimental analyses are presented with four recent algorithms on various continuous problems issued from theory and practice. The proposed method achieves a better performance than previous algorithms, solving all problems with up to one million variables in less than one minute on a modern computer.

Key words. Separable convex optimization, resource allocation, nested constraints, project crashing, speed optimization, lot sizing

AMS subject classifications. 90C25, 52A41, 90B06, 90B35

1. Problem statement. Consider the minimization problem (1.1–1.4), with either integer or continuous variables, where the functions $f_i : [0, d_i] \rightarrow \mathbb{R}$, $i \in \{1, \ldots, n\}$ are proper convex (but not necessarily strictly convex or differentiable); $(s[1], \ldots, s[m])$ is an increasing sequence of $m \geq 1$ integers in $\{1, \ldots, n\}$ such that $s[m] = n$; and the parameters a_i, d_i and B are positive integers:

$$\min \quad f(\mathbf{x}) = \sum_{i=1}^{n} f_i(x_i) \tag{1.1}$$

$$\text{s.t.} \quad \sum_{k=1}^{s[i]} x_k \leq a_i \qquad i \in \{1, \ldots, m-1\} \tag{1.2}$$

$$\sum_{i=1}^{n} x_i = B \tag{1.3}$$

$$0 \leq x_i \leq d_i \qquad i \in \{1, \ldots, n\}. \tag{1.4}$$

To ease the presentation, we define $a_0 = 0$, $a_m = B$, $s[0] = 0$, $\alpha_i = a_i - a_{i-1}$ and $y_i = \sum_{k=1}^{s[i]} x_k$ for $i \in \{1, \ldots, m\}$. Without loss of generality, we can assume that $a_i \leq a_{i+1} \leq a_i + \sum_{k=s[i-1]+1}^{s[i]} d_k$ for $i \in \{1, \ldots, m-1\}$, since a simple lifting of the constraints enables to reformulate the problem in $O(n)$ to fulfill these conditions (see Appendix).

*Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139 (vidalt@mit.edu).

†Department of Electrical Engineering and Computer Science, Laboratory for Information and Decision Systems, Operations Research Center, Massachusetts Institute of Technology, Cambridge, MA 02139 (jaillet@mit.edu).

‡COPPE - Systems Engineering and Computer Science, Federal University of Rio de Janeiro, Rio de Janeiro, RJ 21941-972, Brazil (maculan@cos.ufrj.br).

The problem (1.1–1.4) appears prominently in a variety of applications related to project crashing [41], production and resource planning [4, 5, 44], lot sizing [42], assortment with downward substitution [16, 36, 39], departure-time optimization in vehicle routing [17], vessel speed optimization [32], and telecommunications [33], among many others. When $m = n$, and thus $s[i] = i$ for $i \in \{1, \ldots, n\}$ this problem is known as the resource allocation problem with nested constraints (NESTED). In the remainder of this paper, we maintain the same name for any $m \geq 2$. Without the constraints (1.2), the problem becomes a resource allocation problem (RAP), surveyed in [22, 35], which is the focus of numerous papers related to search-effort allocation, portfolio selection, energy optimization, sample allocation in stratified sampling, capital budgeting, mass advertising, and matrix balancing, among many others. The RAP can also be solved by cooperating agents under some mild assumption on functions f_i [24].

In this paper, we propose efficient polynomial algorithms for both the integer and continuous version of the problem. Computational complexity concepts are well-defined for linear problems. In contrast, with the exception of seminal works such as [18, 20, 29, 31], the complexity of algorithms for general non-linear optimization problems is less discussed in the literature, mostly due to the fact that an infinite output size may be needed due to real optimal solutions. To circumvent this issue, we assume the existence of an oracle which returns the value of $f_i(x)$ for any x in a constant number of operations, and rely on an approximate notion of optimality for non-linear optimization problems [20]. A solution $\mathbf{x}^{(\epsilon)}$ of a continuous problem is ϵ-accurate if and only if there exists an optimal solution $\mathbf{x}^*$ such that $||(\mathbf{x}^{(\epsilon)} - \mathbf{x}^*)||_\infty \leq \epsilon$. This accuracy is defined in the solution space, in contrast with some other approximation approaches which considered the objective space [31].

Two main classes of methods can be found for NESTED, when $m = n$. The algorithms of Padakandla and Sundaresan [33] and Wang [45] can be qualified as *dual*: they solve a succession of RAP subproblems via Lagrange-multiplier optimizations and iteratively re-introduce active nested constraints (1.2). These methods attain a complexity of $O(n^2\Phi_{\text{RAP}}(n, B))$ and $O(n^2 \log n + n\Phi_{\text{RAP}}(n, B))$ for the continuous case, $\Phi_{\text{RAP}}(n, B)$ being the complexity of solving one RAP with n tasks. It should be noted that the performance of the algorithm in [33] can be improved for some continuous problems in which the Lagrangian equation admits a closed and additive form. Otherwise, the RAP are solved by a combination of Newton-Raphson and bisection search on the Lagrangian dual. A more precise computational complexity statement for these algorithms would require a description of the complexity of these sub-procedures and the approximation allowed at each step. Similar methods have also been discussed, albeit with a different terminology, in early production scheduling and lot sizing literature [28].

Another class of methods, that we classify as *primal*, was initially designed for the integer version of the problem, but also applies to the continuous case. These methods take inspiration from greedy algorithms, which consider all feasible increments of one resource, and select the least-cost one. The greedy method is known to converge [11] to the optimum of the integer problem when the constraints determine a polymatroid. Dyer and Walker [10] thus combine the greedy approach with divide-and-conquer using median search, achieving a complexity of $O(n \log n \log^2 \frac{B}{n})$ in the integer case. More recently, Hochbaum [18] combines the greedy algorithm with a scaling approach. An initial problem is solved with large increments, and the increment size is iteratively divided by two to achieve higher accuracy. At each iteration, and for each variable, only one increment from the previous iteration may require an update. Using efficient

feasibility checking methods, NESTED can then be solved in $O(n \log n \log \frac{B}{n})$. The method can also be applied to the general allocation problem as long as the constraints determine a polymatroid [18].

Finally, without constraints (1.2), the RAP can be solved in $O(n \log \frac{B}{n})$ [13, 18]. This complexity is the best possible [18] in the comparison model and the algebraic tree model with operations $+, -, \times, \div$.

2. Contributions. This paper introduces a new algorithm for NESTED, with a complexity of $O(n \log m \log \frac{B}{n})$ in the integer case, and $O(n \log m \log \frac{B}{\epsilon})$ in the continuous case. This is a *dual*-inspired approach, which solves NESTED as a succession of RAP subproblems as in [33, 45]. It is the first method of this kind to attain the same best known complexity as [18] when $m = n$. In addition, the complexity of the proposed method contains a factor $\log m$ instead of $\log n$ as in [18], making it the fastest known method for problems with sparse constraints, for which $\log m = o(\log n)$. In the presence of a quadratic objective, the proposed algorithm attains a complexity of $O(n \log m)$, smaller than the previous complexity of $O(n \log n)$ [19].

Extensive experimental analyses are conducted to compare our method with previous algorithms, using the same testing environment, on eight problem families with n and m ranging from 10 to 1,000,000. In practice, the proposed method demonstrates a better performance than [18] even when $m = n$, possibly due to the use of very simple data structures. The CPU time is much smaller than [33, 45] and the interior point method of MOSEK. All problems with up to one million variables are solved in less than one minute on a modern computer. The method is suitable for large scale problems, e.g. in image processing and telecommunications, or for repeated use when solving combinatorial optimization problems with a resource allocation sub-structure.

Our experiments also show that few nested constraints (1.2) are usually active in optimal solutions for the considered benchmark instances. In fact, we show that the expected number of active constraints grows logarithmically with m for some classes of randomly-generated problems. As a corollary, we also highlight a strongly polynomial algorithm for a significant subset of problems.

3. The proposed algorithm. We propose a recursive decomposition algorithm, which optimally solves a hierarchy of NESTED subproblems by using information obtained at deeper recursions. Any $\textsc{Nested}(v, w)$ subproblem solved in this process (Equations 3.1–3.4) corresponds to a range of variables $(x_{s[v-1]+1}, \ldots, x_{s[w]})$ in the original problem, setting $y_{v-1} = a_{v-1}$ and $y_w = a_w$. At the end of the recursion, $\textsc{Nested}(1, m)$ returns the desired optimal solution for the complete problem.

$$\textsc{Nested}(v,w) \begin{cases} \min \displaystyle\sum_{i=s[v-1]+1}^{s[w]} f_i(x_i) & & (3.1) \\ \text{s.t.} \displaystyle\sum_{k=s[v-1]+1}^{s[i]} x_k \le a_i - a_{v-1} & i \in \{v, \ldots, w-1\} & (3.2) \\ \displaystyle\sum_{i=s[v-1]+1}^{s[w]} x_i = a_w - a_{v-1} & & (3.3) \\ 0 \le x_i \le d_i & i \in \{s[v-1]+1, \ldots, s[w]\} & (3.4) \end{cases}$$

The solution process is illustrated in Figure 3.1 for a problem with $n = 8$ variables and $m = 4$ nested constraints, such that $(s[1], \ldots, s[4]) = (2, 3, 6, 8)$.

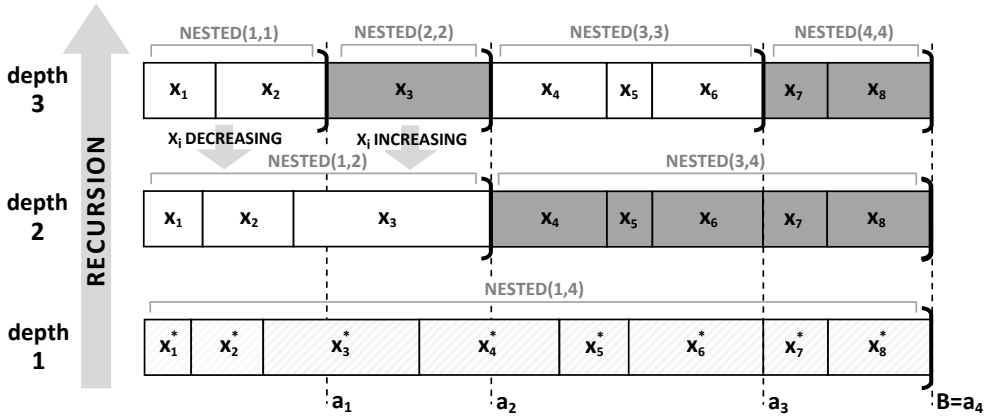

FIG. 3.1. *Main principles of the proposed decomposition algorithm*

At the deepest level of the recursion (depth= 3 on Figure 3.1), each value y_v is fixed to a_v for $v \in \{1, \ldots, m\}$. Equivalently, all nested constraints (symbolized by "]" on the figure) are active. The resulting NESTED(v, v) subproblems, for $v \in \{1, \ldots, m\}$, do not contain any constraints of (3.2), and thus can be solved via classical techniques for RAP (e.g., optimization of the Lagrange multiplier, or the algorithm of [13]).

Higher in the recursion, the ranges of indices corresponding to the NESTED subproblems are iteratively combined, and the nested constraints (3.2) are considered. When solving a NESTED(v, w) subproblem for $v < w$, there exists t such that the optimal solutions of NESTED(v, t) and NESTED$(t + 1, w)$ are known. The efficiency of the decomposition algorithm depends on our ability to use that information in order to simplify NESTED(v, w). To this extent, we introduce an important property of monotonicity on the values of the variables x_i in Theorems 3.1 and 3.2. The proofs are given in the next section.

THEOREM 3.1. *Consider (v, t, w) such that $1 \leq v \leq t < w \leq m$.*
Let $(x^{\downarrow *}_{s[v-1]+1}, \ldots, x^{\downarrow *}_{s[t]})$ and $(x^{\uparrow *}_{s[t]+1}, \ldots, x^{\uparrow *}_{s[w]})$ be optimal solutions of NESTED(v, t) *and* NESTED$(t + 1, w)$ *with integer variables, respectively. Then,* NESTED(v, w) *with integer variables admits an optimal solution $(x^{**}_{s[v-1]+1}, \ldots, x^{**}_{s[w]})$ such that $x^{**}_i \leq x^{\downarrow *}_i$ for $i \in \{s[v-1]+1, \ldots, s[t]\}$ and $x^{**}_i \geq x^{\uparrow *}_i$ for $i \in \{s[t]+1, \ldots, s[w]\}$.*

THEOREM 3.2. *The statement of Theorem 3.1 is also valid for the problem with continuous variables.*

These two theorems guarantee that an optimal solution exists for NESTED(v, w) even in the presence of some additional restrictions on the range of the variables x_i: the values of the variables $(x_{s[v-1]+1}, \ldots, x_{s[t]})$ should not be greater than in the subproblem (in white on Figure 3.1), and the values of the variables $(x_{s[t]+1}, \ldots, x_{s[w]})$ should not be smaller than in the subproblem (in dark gray on Figure 3.1). These valid bounds for the variables x_i can be added to the formulation of NESTED(v, w). Moreover, as stated in Corollary 3.3, these constraints alone guarantee that the nested constraints (3.2) are satisfied:

COROLLARY 3.3. *Let* $(x^{\downarrow *}_{s[v-1]+1}, \ldots, x^{\downarrow *}_{s[t]})$ *and* $(x^{\uparrow *}_{s[t]+1}, \ldots, x^{\uparrow *}_{s[w]})$ *be two optimal solutions of* NESTED(v,t) *and* NESTED$(t+1,w)$*, respectively. Then,* RAP(v,w) *with the coefficients* $\bar{\mathbf{c}}$ *and* $\bar{\mathbf{d}}$ *given below admits at least one optimal solution, and any such optimal solution is also an optimal solution of* NESTED(v,w)*. This proposition is valid for continuous and integer variables.*

$$(\bar{c}_i, \bar{d}_i) = \begin{cases} (0, x_i^{\downarrow *}) & i \in \{s[v-1]+1, \ldots, s[t]\} \\ (x_i^{\uparrow *}, d_i) & i \in \{s[t]+1, \ldots, s[w]\} \end{cases}$$

As a consequence, the range of each variable x_i can be updated and the nested constraints can be eliminated. Each NESTED(v,w) subproblem can then be reduced to a RAP subproblem, as formulated in Equations (3.5–3.7):

$$\text{RAP}(v,w) \begin{cases} \min \displaystyle\sum_{i=s[v-1]+1}^{s[w]} f_i(x_i) & (3.5) \\ \text{s.t.} \displaystyle\sum_{i=s[v-1]+1}^{s[w]} x_i = a_w - a_{v-1} & (3.6) \\ \bar{c}_i \le x_i \le \bar{d}_i \qquad i \in \{s[v-1]+1, \ldots, s[w]\} & (3.7) \end{cases}$$

This leads to a remarkably simple algorithm for NESTED, described in Algorithm 1. The variable ranges are initially set to $\bar{\mathbf{c}} = (0, \ldots, 0)$ and $\bar{\mathbf{d}} = (d_1, \ldots, d_n)$, and NESTED$(1,m)$ is called. At each level of the recursion, an optimal solution to NESTED(v,w) is obtained by solving the two subproblems NESTED(v,t) and NESTED$(t+1,w)$ (Algorithm 1, Lines 5 and 6), along with a modified RAP(v,w) with an updated range for $(x_{s[v-1]+1}, \ldots, x_{s[w]})$ (Algorithm 1, Lines 7 to 11).

Algorithm 1: NESTED(v,w)

```
1  if v = w then
2  |  (x_{s[v-1]+1}, ..., x_{s[v]}) ← RAP(v, v)
3  else
4  |  t ← ⌊(v+w)/2⌋
5  |  (x_{s[v-1]+1}, ..., x_{s[t]}) ← NESTED(v, t)
6  |  (x_{s[t]+1}, ..., x_{s[w]}) ← NESTED(t+1, w)
7  |  for i = s[v-1]+1 to s[t] do
8  |  |  (c̄_i, d̄_i) ← (0, x_i)
9  |  for i = s[t]+1 to s[w] do
10 |  |  (c̄_i, d̄_i) ← (x_i, d_i)
11 |  (x_{s[v-1]+1}, ..., x_{s[w]}) ← RAP(v, w)
```

4. Proof of optimality. We first show that the RAP(v,v) subproblems solved at the deepest level of the recursion admit a feasible solution. We then prove Theorems 3.1 and 3.2 as well as Corollary 3.3. The proof of Theorem 3.1 is based on the optimality properties of a greedy algorithm for resource allocation problems in the presence of constraints forming a polymatroid. The proof of Theorem 3.2 relies on the proximality arguments of [18]. Overall, this demonstrates that the recursive algorithm returns an optimal solution, even for continuous or quadratic problems.

• **RAP(v,v) admits a feasible solution:** A feasible solution can be generated as follows:

for $i = s[v-1]+1$ **to** $s[v]$, $x_i = \min\{d_i, a_v - a_{v-1} - \sum_{k=s[v-1]+1}^{i-1} x_k\}$.

• **Proof of Theorem 3.1.** This proof relies on the optimality of the greedy algorithm for general RAP in the presence of polymatroidal constraints [11, 22]. This greedy algorithm is applied to NESTED(v,w) in Algorithm 2. It iteratively considers all variables x_i which can be feasibly incremented by one unit, and increments the least-cost one. There is one degree of freedom in case of tie (Line 4). In the proof, we add a marginal component in the objective function to break these ties in favor of increments that are part of desired optimal solutions.

Algorithm 2: GREEDY

```
1  x = (x_1, ..., x_n) ← (0, ..., 0)
2  E ← (1, ..., n) ; I ← a_w − a_{v−1}
3  while I > 0 and E ≠ ∅ do
4  |  Find i ∈ E such that f_i(x_{i+1}) − f_i(x_i) = min_{k∈E}{f_k(x_{k+1}) − f_k(x_k)}
5  |  x' ← x ; x'_i ← x_i + 1
6  |  if x' is feasible then
7  |  |  x ← x' ; I ← I − 1
8  |  else
9  |  |  E ← E\{i}
10 if I > 0 then
11 |  return INFEASIBLE
12 else
13 |  return x
```

First, $(x^{\downarrow *}_{s[v-1]+1}, \ldots, x^{\downarrow *}_{s[t]}, x^{\uparrow *}_{s[t]+1}, \ldots, x^{\uparrow *}_{s[w]})$ is a feasible solution of NESTED(v,w), and thus at least one optimal solution $\widetilde{\mathbf{x}}$ of NESTED(v,w) exists. Define $\widetilde{a}_t = a_{v-1} + \sum_{k=s[v-1]+1}^{s[t]} \widetilde{x}_k$. The feasibility of $\widetilde{\mathbf{x}}$ leads to

$$0 \leq \widetilde{a}_t \leq a_t, \tag{4.1}$$

$$\widetilde{a}_t + \sum_{k=s[t]+1}^{s[w]} d_k \geq a_w. \tag{4.2}$$

The problem NESTED(v,t) has a discrete and finite set of solutions, and the associated set of objective values is discrete and finite. If all feasible solutions are optimal, then set $\xi = 1$, otherwise let $\xi > 0$ be the gap between the best and the second best objective value. Consider NESTED(v,t) with a modified separable objective function $\bar{\mathbf{f}}$ such that for $i \in \{s[v-1]+1, \ldots, s[t]\}$,

$$\bar{f}_i(x) = f_i(x) + \frac{\xi}{B+1} \max\{x - x_i^{\downarrow *}, 0\}. \tag{4.3}$$

Any solution $\mathbf{x}$ of the modified NESTED(v,t) with $\bar{\mathbf{f}}$ is an optimal solution of the original problem with $\mathbf{f}$ if and only if $\bar{\mathbf{f}}(\mathbf{x}) < \mathbf{f}(\mathbf{x}^{\downarrow *}) + \xi$, and the new problem admits the unique optimal solution $\mathbf{x}^{\downarrow *}$. Thus, GREEDY returns $\mathbf{x}^{\downarrow *}$ after $a_t - a_{v-1}$ increments. Let $\mathbf{x}^{**} = (x^{**}_{s[v-1]+1}, \ldots, x^{**}_{s[t]})$ be the solution obtained at increment $\widetilde{a}_t - a_{v-1}$. By the properties of GREEDY, $\mathbf{x}^{**}$ is an optimal solution of NESTED(v,t) when replacing

a_t by $\widetilde{a}_t$, such that $x_i^{**} \leq x_i^{\downarrow *}$ for $i \in \{s[v-1]+1, \ldots, s[t]\}$.

The same process can be used for the subproblem NESTED$(t+1, w)$. With the change of variables $\hat{x}_i = d_i - x_i$, and $g_i(x) = f_i(d_i - x)$, the problem becomes

(4.4)

$$\text{NESTED-BIS}(t+1,w) \begin{cases} \min & \displaystyle\sum_{i=s[t]+1}^{s[w]} g_i(\hat{x}_i) & \\ \text{s.t.} & \displaystyle\sum_{k=s[i]+1}^{s[w]} \hat{x}_k \leq a_i - a_w + \sum_{k=s[i]+1}^{s[w]} d_k & i \in \{t+1, \ldots, w-1\} \\ & \displaystyle\sum_{k=s[t]+1}^{s[w]} \hat{x}_k = a_t - a_w + \sum_{k=s[t]+1}^{s[w]} d_k & \\ & 0 \leq \hat{x}_i \leq d_i & i \in \{s[t]+1, \ldots, s[w]\}. \end{cases}$$

If all feasible solutions of NESTED-BIS$(t+1, w)$ are optimal, then set $\hat{\xi} = 1$, otherwise let $\hat{\xi} > 0$ be the gap between the best and second best solution of NESTED-BIS$(t+1, w)$. For $i \in \{s[t]+1, \ldots, s[w]\}$, define $\hat{x}_i^{\uparrow *} = d_i - x_i^{\uparrow *}$ and $\bar{\mathbf{g}}$ such that

$$\bar{g}_i(x) = g_i(x) + \frac{\hat{\xi}}{B+1} \max\{x - \hat{x}_i^{\uparrow *}, 0\}. \tag{4.5}$$

GREEDY returns $\hat{\mathbf{x}}^{\uparrow *}$, the unique optimal solution of NESTED-BIS$(t+1, w)$ with the modified objective $\bar{\mathbf{g}}$. Let $\hat{\mathbf{x}}^{**}$ be the solution obtained at step $\widetilde{a}_t - a_w + \sum_{k=s[t]+1}^{s[w]} d_k$. This step is non-negative according to Equation (4.2). GREEDY guarantees that $\hat{\mathbf{x}}^{**}$ is an optimal solution of NESTED-BIS$(t+1, w)$ with the alternative equality constraint

$$\sum_{k=s[t]+1}^{s[w]} \hat{x}_k = \widetilde{a}_t - a_w + \sum_{k=s[t]+1}^{s[w]} d_k. \tag{4.6}$$

In addition, $\hat{x}_i^{**} \leq \hat{x}_i^{\uparrow *}$ for $i \in \{s[t]+1, \ldots, s[w]\}$. Reverting the change of variables, this leads to an optimal solution $\mathbf{x}^{**}$ of NESTED$(t+1, w)$ where a_t has been replaced by $\widetilde{a}_t$, and such that $x_i^{**} \geq x_i^{\uparrow *}$ for $i \in \{s[t]+1, \ldots, s[w]\}$.

Overall, since $\mathbf{x}^{**}$ is such that $\sum_{k=s[v-1]+1}^{s[t]} x_k^{**} = \sum_{k=s[v-1]+1}^{s[t]} \widetilde{x}_k = \widetilde{a}_t - a_{v-1}$, since it is also optimal for the two subproblems obtained when fixing $\widetilde{a}_t = a_{v-1} + \sum_{k=s[v-1]+1}^{s[t]} x_k^{**}$, then x^{**} is an optimal solution of NESTED(v, w) which satisfies the requirements of Theorem 3.1.

• **Proof of Theorem 3.2.** The proof relies on the proximity theorem of Hochbaum [18] for general resource allocation problem with polymatroidal constraints. This theorem states that for any optimal continuous solution $\mathbf{x}$ there exists an optimal solution $\mathbf{z}$ of the same problem with integer variables, such that $\mathbf{z} - \mathbf{e} < \mathbf{x} < \mathbf{z} + n\mathbf{e}$, and thus $||\mathbf{z} - \mathbf{x}||_\infty \leq n$. Reversely, for any integer optimal solution $\mathbf{z}$, there exists an optimal continuous solution such that $||\mathbf{z} - \mathbf{x}||_\infty \leq n$.

Let $(x_{s[v-1]+1}^{\downarrow *}, \ldots, x_{s[t]}^{\downarrow *})$ and $(x_{s[t]+1}^{\uparrow *}, \ldots, x_{s[w]}^{\uparrow *})$ be two optimal solutions of NESTED(v, t) and NESTED$(t+1, w)$ with continuous variables, and suppose that

the statement of Theorem 3.1 is false for the continuous case. Hence, there exists $\Delta > 0$ such that for any optimal solution $\mathbf{x}^{**}$ of the continuous NESTED(v, w) there exists either $i \in \{s[v-1]+1, \ldots, s[t]\}$ such that $x_i^{**} \geq \Delta + x_i^{\downarrow *}$, or $i \in \{s[t]+1, \ldots, s[w]\}$ such that $x_i^{**} \leq x_i^{\uparrow *} - \Delta$. We will prove that this statement is impossible.

Define the scaled problem NESTED-$\beta(v, t)$ below. This problem admits at least one feasible integer solution as a consequence of the feasibility of NESTED(v, t).

(4.7)

$$\text{NESTED-}\beta(v,t) \begin{cases} \min & \displaystyle\sum_{i=s[v-1]+1}^{s[t]} f_i\left(\frac{x_i}{\beta}\right) & \\ \text{s.t.} & \displaystyle\sum_{k=s[v-1]+1}^{s[i]} x_k \leq \beta a_i - \beta a_{v-1} & i \in \{v, \ldots, t-1\} \\ & \displaystyle\sum_{i=s[v-1]+1}^{s[t]} x_i = \beta a_t - \beta a_{v-1} & \\ & 0 \leq x_i \leq \ \beta d_i & i \in \{s[v-1]+1, \ldots, s[t]\} \end{cases}$$

The proximity theorem of [18] guarantees the existence of a serie of integer solutions $\hat{\mathbf{x}}^{\downarrow *[\beta]}$ of NESTED-$\beta(v, t)$ such that $\lim_{\beta \to \infty} \|\hat{\mathbf{x}}^{\downarrow *[\beta]}/\beta - \mathbf{x}^{\downarrow *}\| = 0$. With the same arguments, the existence of a serie of integer solutions $\hat{\mathbf{x}}^{\uparrow *[\beta]}$ of NESTED-$\beta(t+1, w)$ such that $\lim_{\beta \to \infty} \|\hat{\mathbf{x}}^{\uparrow *[\beta]}/\beta - \mathbf{x}^{\uparrow *}\| = 0$ is also demonstrated.

As a consequence of Theorem 3.1, for any β there exists an integer optimal solution $\hat{\mathbf{x}}^{**[\beta]}$ of NESTED-$\beta(v, w)$, such that $\hat{x}_i^{**[\beta]} \leq \hat{x}_i^{\downarrow *[\beta]}$ for $i \in \{s[v-1]+1, \ldots, s[t]\}$ and $\hat{x}_i^{**[\beta]} \geq \hat{x}_i^{\uparrow *[\beta]}$ for $i \in \{s[t]+1, \ldots, s[w]\}$.

Finally, the proximity theorem of [18] guarantees the existence of continuous solutions $\mathbf{x}^{**[\beta]}$ of NESTED-$\beta(v, w)$, such that $\lim_{\beta \to \infty} \|\mathbf{x}^{**[\beta]} - \hat{\mathbf{x}}^{**[\beta]}/\beta\| = 0$. Hence, there exist β, $\hat{\mathbf{x}}^{\downarrow *[\beta]}$, $\hat{\mathbf{x}}^{\uparrow *[\beta]}$, $\hat{\mathbf{x}}^{**[\beta]}$ and $\mathbf{x}^{**[\beta]}$ such that $\|\hat{\mathbf{x}}^{\downarrow *[\beta]}/\beta - \mathbf{x}^{\downarrow *}\| \leq \Delta/3$, $\|\hat{\mathbf{x}}^{\uparrow *[\beta]}/\beta - \mathbf{x}^{\uparrow *}\| \leq \Delta/3$, and $\|\hat{\mathbf{x}}^{**[\beta]}/\beta - \mathbf{x}^{**[\beta]}\| \leq \Delta/3$.

For $i \in \{s[v-1]+1, \ldots, s[t]\}$, we thus have:

$$x_i^{**[\beta]} \leq \frac{\hat{x}_i^{**[\beta]}}{\beta} + \frac{\Delta}{3} \leq \frac{\hat{x}_i^{\downarrow *[\beta]}}{\beta} + \frac{\Delta}{3} \leq x_i^{\downarrow *} + \frac{2\Delta}{3}.$$

As a consequence, the statement $x_i^{**[\beta]} \geq \Delta + x_i^{\downarrow *}$ is false.

For $i \in \{s[t]+1, \ldots, s[w]\}$, we thus have:

$$x_i^{**[\beta]} \geq \frac{\hat{x}_i^{**[\beta]}}{\beta} - \frac{\Delta}{3} \geq \frac{\bar{x}_i^{\uparrow *[\beta]}}{\beta} - \frac{\Delta}{3} \geq x_i^{\uparrow *} - \frac{2\Delta}{3}.$$

As a consequence, the statement $x_i^{**[\beta]} \leq x_i^{\uparrow *} - \Delta$ is false, and the solution $x^{**[\beta]}$ leads to the announced contradiction.

4.1. Proof of Corollary 3.3. As demonstrated in Theorems 3.1 and 3.2, there exists an optimal solution $\mathbf{x}^{**}$ of NESTED(v,w), such that $x_k^{**} \leq x_k^{\downarrow *}$ for $k \in \{s[v-1]+1,\ldots,s[t]\}$ and $x_k^{**} \geq x_k^{\uparrow *}$ for $k \in \{s[t]+1,\ldots,s[w]\}$. These two sets of constraints can be introduced in the formulation (3.1–3.4). Any optimal solution of this strengthened formulation is an optimal solution of NESTED(v,w), and the strengthened formulation admits at least one feasible solution. The following relations hold for any solution $\mathbf{x} = (x_{s[v-1]+1},\ldots,x_{s[w]})$:

$$
\begin{aligned}
& x_k \leq x_k^{\downarrow *} \text{ for } k \in \{s[v-1]+1,\ldots,s[t]\} \\
& \qquad \Rightarrow \sum_{k=s[v-1]+1}^{s[i]} x_k \leq \sum_{k=s[v-1]+1}^{s[i]} x_k^{\downarrow *} \\
& \qquad \Rightarrow \sum_{k=s[v-1]+1}^{s[i]} x_k \leq a_i - a_{v-1} \qquad \text{for } i \in \{v,\ldots,t\}
\end{aligned} \tag{4.8}
$$

$$
\begin{aligned}
& x_k \geq x_k^{\uparrow *} \text{ for } k \in \{s[t]+1,\ldots,s[w]\} \\
& \qquad \Rightarrow \sum_{k=s[i]+1}^{s[w]} x_k \geq \sum_{k=s[i]+1}^{s[w]} x_k^{\uparrow *} \\
& \qquad \Rightarrow \sum_{k=s[v-1]+1}^{s[i]} x_k \leq \sum_{k=s[v-1]+1}^{s[i]} x_k^{\uparrow *} \\
& \qquad \Rightarrow \sum_{k=s[v-1]+1}^{s[i]} x_k \leq a_i - a_{v-1} \qquad \text{for } i \in \{t,\ldots,w-1\}
\end{aligned} \tag{4.9}
$$

Hence, any solution satisfying the constraints $x_i \leq \bar{d}_i$ for $i \in \{s[v-1]+1,\ldots,s[t]\}$ and $x_i \geq \bar{c}_i$ for $i \in \{s[t]+1,\ldots,s[w]\}$ also satisfies the constraints of Equation (3.2). These constraints can thus be removed, leading to the formulation RAP(v,w).

5. Computational complexity. This section investigates the computational complexity of the proposed method for integer and continuous problems, as well as for the specific case of quadratic objective functions.

THEOREM 5.1. *The proposed algorithm for NESTED with integer variables works with a complexity of* $O(n \log m \log \frac{B}{n})$.

Proof. The integer NESTED problem is solved as a hierarchy of RAP, with $h = 1 + \lceil \log_2 m \rceil$ levels of recursion (Algorithm 1, Lines 4–6). At each level $i \in \{1,\ldots,h\}$, 2^{h-i} RAP subproblems are solved (Algorithm 1, Lines 2 and 11). Furthermore, $O(n)$ operations per level are needed to update $\bar{\mathbf{c}}$ and $\bar{\mathbf{d}}$ (Algorithm 1, Lines 7–10). The method of Frederickson and Johnson [13] for RAP works in $O(n \log \frac{B}{n})$. Hence, each RAP(v,w) can be solved in $O((s[w]-s[v]) \log \frac{a_w - a_v}{s[w]-s[v]})$ operations. Overall, there exist positive constants K, K' and K'' such that the number of operations $\Phi(n,m)$ of the proposed method is:

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

This leads to the announced complexity of $O(n\log m\log\frac{B}{n})$. □

For the continuous case, two situations can be considered. When there exists an "exact" solution method independent of ϵ to solve the RAP subproblems, e.g. when the objective function is quadratic, the convergence is guaranteed by Theorem 3.2. As such, the algorithm of Brucker [7] or Maculan et al. [25] can be used to solve each quadratic RAP subproblem in $O(n)$, leading to an overall complexity of $O(n\log m)$ to solve the quadratic NESTED resource allocation problem.

In the more general continuous case without any other assumption on the objective functions, all problem parameters can be scaled by a factor $\frac{n}{\epsilon}$ [18], and the integer problem with $B' = \frac{Bn}{\epsilon}$ can be be solved with complexity $O(n\log m\log\frac{B'}{n\epsilon})$ $= O(n\log m\log\frac{B}{\epsilon})$. The proximity theorem guarantees that an ϵ-accurate solution of the continuous problem is obtained after the reverse change of variables.

Finally, we have assumed in this paper integer values for a_i, B, and d_i. Now, consider fractional parameter values with z significant figures and x decimal places. All problem coefficients as well as ϵ can be scaled by a factor 10^x to obtain integer parameters, and the number of elementary operations of the method remains the same. We assume in this process that operations are still elementary for numbers of $z+x$ digits. This is a common assumption when dealing with continuous parameters.

6. Experimental analyses. Few detailed computational studies on nested resource allocation problems can be found in previous works. The theoretical complexity of the algorithm of Hochbaum [18] was investigated, but not its experimental performance. The algorithms of [33] and [45] were originally implemented in Matlab, possibly leading to higher CPU times. So, in order to assess the practical performance of all algorithms on a fair common basis, we implemented all of the most recent ones using the same language (C++). These algorithms are very simple, concise, and require similar array data structures and elementary arithmetics, hence limiting possible bias related to programming style or implementation skills. In particular, we have compared:

- PS09 : the dual algorithm of Padakandla and Sundaresan [34];
- W14 : the dual algorithm of Wang [45];
- H94 : the scaled greedy algorithm of Hochbaum [18];
- MOSEK : the interior point method of MOSEK [1, for conic quadratic opt.];
- THIS : our proposed method.

Note that the new implementations of PS09 and W14 become (10 to 100 times) faster than their original Matlab implementations.

Each algorithm is tested on NESTED instances with three types of objective functions. The first objective function profile comes from [34, 45]. We also consider two other objectives related to project and production scheduling applications. The size of instances ranges from $n = 10$ to $1,000,000$. To further investigate the impact of the number of nested constraints, additional instances with a fixed number of tasks and a variable number of nested constraints are also considered. An accuracy of $\epsilon = 10^{-8}$ is sought, and all tests are conducted on a single Xeon 3.07 GHz CPU, with a single thread. To obtain accurate time measurements, any algorithm with a run time smaller than one second has been executed multiple times in a loop. In this case, we report the total CPU time divided by the number of runs.

The instances proposed in [34, 45] are continuous with non-integer parameters a_i and B. We generated these parameters with nine decimals. Following the last remark of Section 5, all problem parameters and ϵ can be multiplied by 10^9 to obtain a problem with integer coefficients. For a fair comparison with previous authors, we rely on a similar RAP method as [34, 40, 45], using bisection search on the Lagrangian equation to solve the subproblems. The derivative f_i' of f_i is well-defined for all test instances. This Lagrangian method does not have the same complexity guarantees as [13], but performs reasonably well in practice. The initial bisection-search interval for each $\textsc{Rap}(v, w)$ is set to $[\min_{i\in\{s[v-1]+1,\ldots,s[w]\}} f_i'(\bar{c}_i), \max_{i\in\{s[v-1]+1,\ldots,s[w]\}} f_i'(\bar{d}_i)]$.

Implementing previous algorithm from the literature led to a few other questions and implementation choices. As mentioned in [18], the Union-Find structure of [15] achieves a $O(1)$ amortized complexity for feasibility checks. Yet, its implementation is intricate, and we privileged a more standard Union-Find with balancing and path compression [43], attaining a complexity of $\alpha^{\text{ACK}}(n)$ where α^{ACK} is the inverse of the Ackermann function. For all practical purposes, this complexity is nearly constant and the practical performance of the simpler implementation is very similar [15]. The algorithm of [18] also requires a correction, which is well-documented in [30].

Finally, as discussed in Section 6.1, the distributions and ordering of the parameters, in previous benchmark instances, led to optimal solutions with few active nested constraints. To investigate the performance of all methods on a larger range of settings, we completed the benchmark with other parameter distributions, leading to five sets of instances:

6.1. Problem instances – previous literature. We first consider the test function (6.1) from [34]. The problem was originally formulated with nested constraints of the type $\sum_{k=1}^{s[i]} x_k \geq a_i$ for $i \in \{1, \ldots, m-1\}$. The change of variables $\hat{x_i} = 1 - x_i$ can be used to obtain (1.1–1.4).

$$\text{(6.1)} \qquad \text{[F]} \qquad f_i(x) = \frac{x^4}{4} + p_i x, \qquad\qquad x \in [0, 1]$$

The benchmark instances of [34] have been generated with uniformly distributed p_i and α_i in [0,1] (recall that $\alpha_i = a_i - a_{i-1}$). The parameters p_i are then ordered by increasing value. As observed in our experiments, this ordering of parameters leads to very few active nested constraints. We thus introduce two additional instance sets called [F-Uniform] and [F-Active]. In [F-Uniform], the parameters p_i and α_i are generated with uniform distribution, between [0,1] and [0,0.5], respectively, and non-ordered. [F-Active] is generated in the same way, and α_i are sorted in decreasing order. As a consequence, these latter instances have many active constraints. In [34],

some other test functions were considered, for which the solutions of the Lagrangian equations admit a closed additive form. As such, each Lagrangian equation can be solved in amortized $O(1)$ instead of $O(n \log \frac{B}{n})$. This technique is specific to such functions and cannot be applied with arbitrary bounds d_i. We thus selected the functions [F] for our experiments as they represent a more general case.

6.2. Problem instances – Project crashing. A seminal problem in project management [23, 27] relates to the optimization of a critical path of tasks in the presence of non-linear cost/time trade-off functions $f_i(x)$, expressing the cost of processing a task i in x_i time units. Different types of trade-off functions have been investigated in the literature [6, 8, 12, 14, 37]. The algorithm of this paper can provide the best compression of a critical path to finish a project at time B, while imposing additional deadline constraints $\sum_{k=1}^{s[i]} x_k \leq a_i$ for $i \in \{1, \ldots, m-1\}$ on some steps of the project. Lower and upper bounds on task durations $c_i \leq x_i \leq d_i$ are also commonly imposed. The change of variables $\hat{x}_i = x_i + c_i$ leads to the formulation (1.1–1.4). Computational experiments are performed on these problems with the cost/time trade-off functions of Equation (6.2), proposed in [12], in which the cost supplement related to crashing grows as the inverse of task duration.

$$\text{[Crashing]} \qquad f_i(x) = k_i + \frac{p_i}{x}, \quad x \in [c_i, d_i] \tag{6.2}$$

Parameters p_i, d_i and α_i are generated by exponential distributions of mean $\mathbb{E}(p_i) = \mathbb{E}(d_i) = 1$ and $\mathbb{E}(\alpha_i) = 0.75$. Finally, $a_i = \sum_{k=1}^{i} \alpha_k$ and $c_i = \min(\alpha_i, \frac{d_i}{2})$ to ensure feasibility.

6.3. Problem instances – Vessel speed optimization. Some applications require solving multiple NESTED problems. One such case relates to an emergent class of vehicle routing and scheduling problems aiming at jointly optimizing vehicle speeds and routes to reach delivery locations within specified time intervals [3, 21, 32]. Heuristic and exact methods for such problems consider a very large number of alternative routes (permutations of visits) during the search. For each route, determining the optimal travel times $(x_1, \ldots, x_n)$ on n trip segments to satisfy m deadlines $(a_1, \ldots, a_m)$ on some locations is the same subproblem as in formulation (1.1–1.4). We generate a set of benchmark instances for this problem, assuming as in [38] that fuel consumption is approximately a cubic function of speed on relevant intervals. In Equation (6.3), p_i is the fuel consumption on the way to location i per time unit at maximum speed, and c_i is the minimum travel time.

$$\text{[FuelOpt]} \qquad f_i(x) = p_i \times c_i \times \left(\frac{c_i}{x}\right)^3, \quad x \in [c_i, d_i] \tag{6.3}$$

Previous works on the topic [21, 32] assumed identical p_i on all edges. Our work allows to raise this simplifying assumption, allowing to take into consideration edge-dependent factors such as currents, water depth, or wind which have a strong impact on fuel consumption. We generate uniform p_i values in the interval $[0.8, 1.2]$. Base travel times c_i are generated with uniform distribution in $[0.7, 1]$, $d_i = c_i * 1.5$, and α_i are generated in $[1, 1.2]$.

6.4. Experiments with m = n. The first set of experiments involves as many nested constraints as variables ($n = m$). We tested the five methods for $n \in \{10, 20, 50, 100, 200, \ldots, 10^6\}$, with 100 different problem instances for each size $n \leq 10,000$, and 10 different problem instances when $n > 10,000$. A time limit of 10 minutes per run

TABLE 6.1
CPU time(s) of five different algorithms for NESTED, with increasing n and $m = n$

Instances	n	nb Active	Time (s)				
			PS09	**W14**	**H94**	**MOSEK**	**THIS**
[F]	10	1.15	8.86×10^{-5}	8.06×10^{-5}	6.18×10^{-5}	8.73×10^{-3}	$\mathbf{1.85\times10^{-5}}$
	10^2	1.04	7.96×10^{-3}	7.03×10^{-3}	6.74×10^{-4}	2.03×10^{-2}	$\mathbf{1.69\times10^{-4}}$
	10^3	1.08	9.17×10^{-1}	7.87×10^{-1}	8.74×10^{-3}	9.63	$\mathbf{1.98\times10^{-3}}$
	10^4	1.15	1.06×10^{2}	8.72×10^{1}	1.46×10^{-1}	–	$\mathbf{2.23\times10^{-2}}$
	10^5	1.20	–	–	2.93	–	$\mathbf{3.67\times10^{-1}}$
	10^6	1.10	–	–	4.42×10^{1}	–	**4.36**
[F-Uniform]	10	2.92	1.03×10^{-4}	4.57×10^{-5}	5.86×10^{-5}	8.76×10^{-3}	$\mathbf{2.62\times10^{-5}}$
	10^2	5.06	1.37×10^{-2}	1.61×10^{-3}	7.42×10^{-4}	2.14×10^{-2}	$\mathbf{4.97\times10^{-4}}$
	10^3	7.65	2.28	8.35×10^{-2}	9.83×10^{-3}	8.63	$\mathbf{8.41\times10^{-3}}$
	10^4	9.99	–	6.08	1.67×10^{-1}	–	$\mathbf{1.31\times10^{-1}}$
	10^5	12.00	–	–	3.99	–	**2.74**
	10^6	14.50	–	–	7.06×10^{1}	–	$\mathbf{4.62\times10^{1}}$
[F-Active]	10	3.67	1.19×10^{-4}	3.94×10^{-5}	5.76×10^{-5}	8.71×10^{-3}	$\mathbf{2.88\times10^{-5}}$
	10^2	10.00	2.28×10^{-2}	9.65×10^{-4}	7.50×10^{-4}	2.18×10^{-2}	$\mathbf{4.69\times10^{-4}}$
	10^3	22.58	4.88	3.82×10^{-2}	9.93×10^{-3}	1.01×10^{1}	$\mathbf{6.81\times10^{-3}}$
	10^4	50.75	–	2.31	1.62×10^{-1}	–	$\mathbf{9.95\times10^{-2}}$
	10^5	114.50	–	2.62×10^{2}	3.18	–	**1.47**
	10^6	280.30	–	–	5.65×10^{1}	–	$\mathbf{2.21\times10^{1}}$
[Crashing]	10	6.44	4.49×10^{-5}	1.81×10^{-5}	5.02×10^{-5}	9.46×10^{-3}	$\mathbf{8\times10^{-6}}$
	10^2	24.61	6.03×10^{-3}	7.05×10^{-4}	6.80×10^{-4}	5.95×10^{-2}	$\mathbf{1.25\times10^{-4}}$
	10^3	34.14	1.10	4.84×10^{-2}	8.86×10^{-3}	1.43×10^{1}	$\mathbf{2.48\times10^{-3}}$
	10^4	46.90	2.50×10^{2}	2.85	1.50×10^{-1}	–	$\mathbf{4.93\times10^{-2}}$
	10^5	50.30	–	2.98×10^{2}	3.44	–	**1.13**
	10^6	88.30	–	–	6.02×10^{1}	–	$\mathbf{2.35\times10^{1}}$
[FuelOpt]	10	2.93	8.46×10^{-5}	3.17×10^{-5}	6.62×10^{-5}	8.74×10^{-3}	$\mathbf{2.20\times10^{-5}}$
	10^2	5.31	1.22×10^{-2}	1.28×10^{-3}	7.98×10^{-4}	1.99×10^{-2}	$\mathbf{4.21\times10^{-4}}$
	10^3	6.86	1.74	7.10×10^{-2}	1.07×10^{-2}	7.02	$\mathbf{6.83\times10^{-3}}$
	10^4	9.53	2.43×10^{2}	4.81	1.95×10^{-1}	–	$\mathbf{1.02\times10^{-1}}$
	10^5	14.90	–	4.34×10^{2}	4.88	–	**1.72**
	10^6	12.80	–	–	8.54×10^{1}	–	$\mathbf{2.99\times10^{1}}$

was imposed. The CPU time of each method for a subset of size values is reported in Table 6.1. The first two columns report the instance set identifier, the next column displays the average number of active constraints in the optimal solutions, and the five next columns report the average run time of each method on each set. The smallest CPU time is highlighted in boldface. A sign "–" means that the time limit is attained without returning a solution. The complete results, for all values of n, are also represented on a logarithmic scale in Figure 6.1.

First, it is remarkable that the number of active nested constraints strongly varies from one set of benchmark instances to another. One drawback of the previously-used [F] instances [34] is that they lead to a low number of active nested constraints, in such a way that in many cases an optimal RAP solution obtained by relaxing all nested constraints is also the optimal NESTED solution. Some algorithms can benefit from such problem characteristics.

The five considered methods require very different CPU time to reach the optimal solution with the same precision. In all cases, the smallest time was achieved by our decomposition method. The time taken by PS09, W14, H94 and our decomposition algorithm, as a function of n, is in most most cases in accordance with the theoretical complexity, cubic for PS09, quadratic for W14, and log-linear for H94 and the proposed method (Figure 6.1). The only notable exception is problem type [F], for which the reduced number of active constraints leads to a general quadratic behavior of PS09

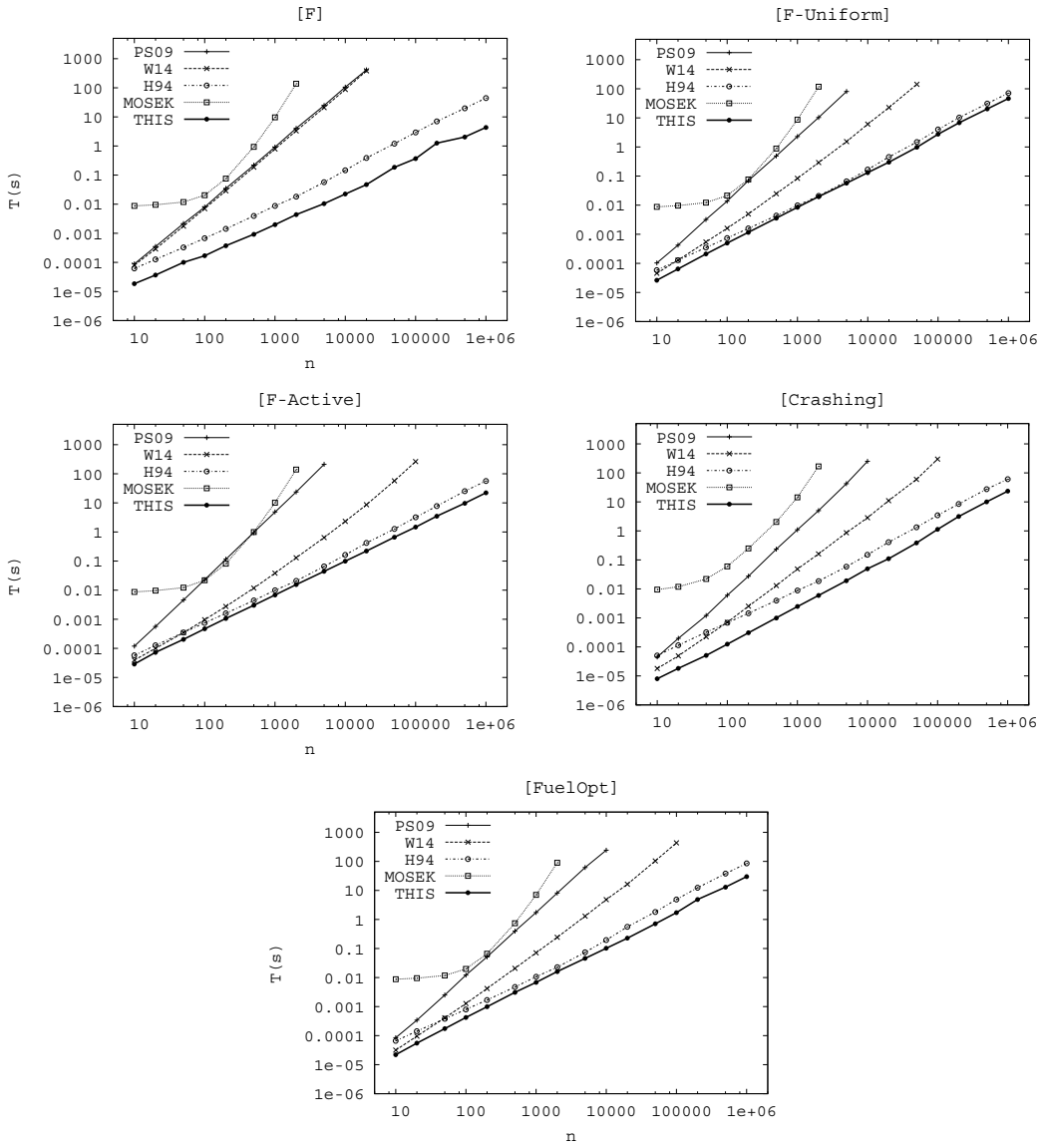

FIG. 6.1. *CPU Time(s) as a function of* $n \in \{10, \ldots, 10^6\}$. $m = n$. *Logarithmic representation*

(instead of cubic). The CPU time of MOSEK does not exhibit a polynomial behavior on the considered problem-size range, possibly because of the preprocessing phase. The proposed method and H94 have a similar growth when $m = n$. Our dual-inspired decomposition algorithm appears to be slightly faster in practice, by a constant factor $\times 1$ to $\times 10$. This may be explained by the use of simpler array data structures (hidden constants related to the use of priority lists or Union-Find data structures are avoided). The bottleneck of our method, measured by means of a time profiler, is the call to the oracle for the objective function. In our implementation of H94, the call to the oracle and the management of the priority list for finding the minimum cost increment contribute equally to the largest part of the CPU time. The time taken by

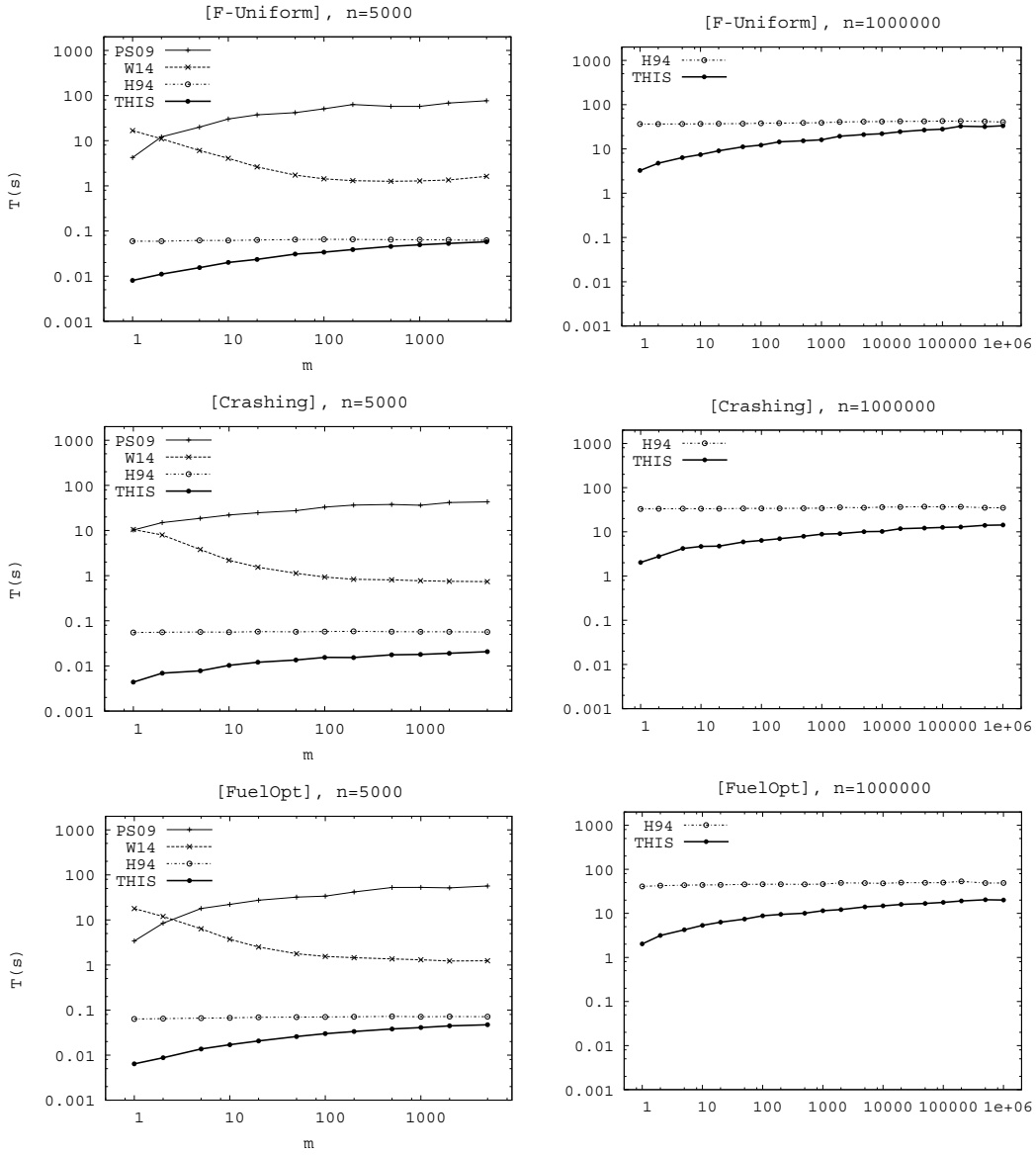

FIG. 6.2. *CPU Time(s) as a function of m.* $n \in \{5000, 1000000\}$. *Logarithmic representation*

the Union-Find structures is not significant.

6.5. Experiments with m < n. In a second set of experiments, the number of variables is fixed and the impact of the number of nested constraints is evaluated, with $m \in \{1, 2, 5, 10, 50, \ldots, n\}$, on [F-Uniform], [Crashing] and [FuelOpt]. Two values $n = 5000$ and $n = 1,000,000$ were considered, to allow experiments with H94, PS09, W14 and the proposed method on medium size problems in reasonable CPU time, as well as further tests with H94 and the proposed method on large-scale instances. The CPU time as a function of m is displayed in Figure 6.2.

The CPU time of H94 appears to be independent of m, while significant time

gains can be observed for the proposed method, which is $\times 5$ to $\times 20$ faster than H94 on large-scale instances ($n = 1,000,000$) with few nested constraints ($m = 10$ or 100). It also appears that PS09 benefits from sparser constraints. Surprisingly, sparser constraints are detrimental to W14 in practice, possibly because Equation (21) of [45] is called on larger sets of variables.

7. A note on the number of active nested constraints. The previous experiments have shown that the number of active nested constraints in the optimal solutions tends to grow sub-linearly for the considered problems. In Table 6.1 for example, even when $m = 10^6$ the number of active nested constraints is located between 12.8 and 88.3 for instances with randomly generated coefficients (no ordering as in [F] or [F-Active]). To complement this observation, we show in the following that the expected number of active nested constraints in a random optimal solution grows logarithmically with m when :

1. $d_i = +\infty$;
2. parameters α_i are outcomes of i.i.d. random variables;
3. functions f_i are strictly convex and differentiable;
4. and there exists a function h and $\gamma_i \in \mathbb{R}^{+*}$ for $i \in \{1, \ldots, n\}$ satisfying $f_i(x) = \gamma_i h(x/\gamma_i)$. γ_i are i.i.d. random variables independent from the α_i's, and the vectors (γ_i, α_i) are non-collinear.

Function shapes satisfying condition 4. are frequently encountered, e.g. in

- crashing: $f_i(x) = p_i/x \Rightarrow h(x) = 1/x$ and $\gamma_i = \sqrt{p_i}$;
- fuel optimization: $f_i(x) = p_i c_i (c_i/x)^3 \Rightarrow h(x) = 1/x^3$ and $\gamma_i = c_i \sqrt[4]{p_i}$;
- any function $f_i(x) = p_i x^k$ s.t. $k \neq 1 \Rightarrow h(x) = x^k$ and $\gamma_i = 1/p_i^{1/(k-1)}$.

The first order necessary and sufficient optimality conditions of problem (1.1–1.3) with $x_i \in \mathbb{R}^+$ for $i \in \{1, \ldots, n\}$ can be written as:

(7.1) $\mathbf{x} = (x_1, \ldots, x_n) \geq \mathbf{0}$ satisfy constraints (1.2) and (1.3)

(7.2) for $i \in \{1, \ldots, m\}$ and $j \in \{s[i-1]+1, \ldots, s[i]-1\}$, $f_j'(x_j) = f_{j+1}'(x_{j+1})$

(7.3) for $i \in \{1, \ldots, m-1\}$ and $j = s[i]$, $\begin{cases} \textbf{either } f_j'(x_j) = f_{j+1}'(x_{j+1}) \\ \textbf{or } f_j'(x_j) < f_{j+1}'(x_{j+1}) \textbf{ and } \sum_{k=1}^{j} x_k = a_i \end{cases}$

If $f_i(x) = \gamma_i h(\frac{x}{\gamma_i})$, then $f_i'(x) = h'(\frac{x}{\gamma_i})$, and with the strict convexity the necessary and sufficient conditions (7.2) and (7.3) become:

(7.2b) for $i \in \{1, \ldots, m\}$ and $j \in \{s[i-1]+1, \ldots, s[i]-1\}$, $\dfrac{x_j}{\gamma_j} = \dfrac{x_{j+1}}{\gamma_{j+1}}$

(7.3b) for $i \in \{1, \ldots, m-1\}$ and $j = s[i]$, $\begin{cases} \textbf{either } \frac{x_j}{\gamma_j} = \frac{x_{j+1}}{\gamma_{j+1}} \\ \textbf{or } \frac{x_j}{\gamma_j} < \frac{x_{j+1}}{\gamma_{j+1}} \textbf{ and } \sum_{k=1}^{j} x_k = a_i \end{cases}$

Define $\Gamma_i = \sum_{k=1}^{i} \gamma_k$ for $i \in \{0, \ldots, n\}$. As illustrated on Figure 7.1, searching for a solution satisfying (1.2), (1.3), (7.2b) and (7.3b) reduces to computing the convex hull of the set of points $\mathcal{P}$ such that

$$\mathcal{P} = \{(\Gamma_{s[j]}, a_j) \mid j \in \{0, \ldots, m\}\}. \tag{7.4}$$

Let $\Phi : [0, \Gamma_n] \rightarrow [0, B]$ be the curve associated with the lower part of the convex hull, in boldface on Figure 7.1. Then, the solution defined as $x_i = \Phi(\Gamma_i) - \Phi(\Gamma_{i-1})$ for $i \in \{1, \ldots, n\}$ satisfies all previously-mentioned conditions since

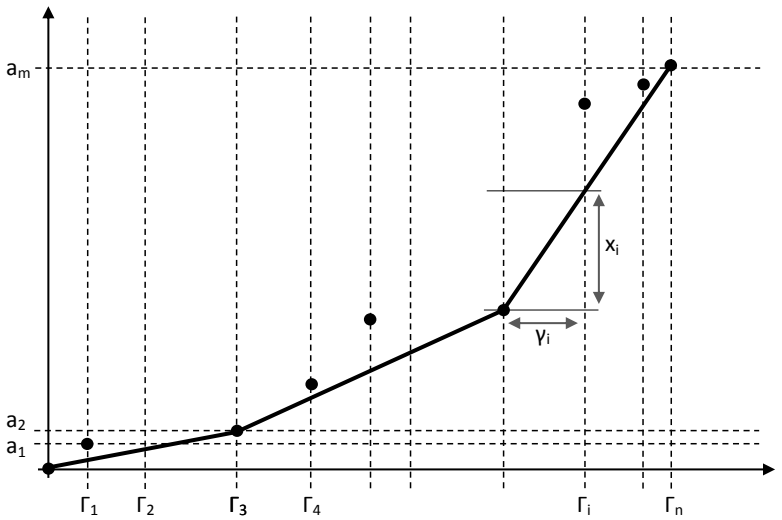

FIG. 7.1. *Reduction of NESTED to a convex hull computation. Example with* $n = 10$, $m = 8$ *and* $s = (1, 3, 4, 5, 7, 8, 9)$.

- Φ is below the points p_j, hence satisfying (1.2);
- p_m is part of the convex hull, thus satisfying (1.3);
- $\Phi(z) \geq 0$ for $z \in [0, \Gamma_n]$ since all p_j coordinates are non-negative, hence $\mathbf{x} \geq \mathbf{0}$;
- the slope of Φ is constant between vertices of the convex hull (7.2b);
- and the slope of Φ only increases when meeting a vertex (7.3b).

The expected number of vertices of a convex hull with random points is at the core of an extensive literature. We refer to [9] for early studies, and [26] for a recent review. Consider a randomly-generated NESTED problem, such that γ_j for $j \in \{1, \ldots, m\}$ and α_j for $j \in \{1, \ldots, n\}$ are i.i.d. random variables. If the distribution is such that all vectors (γ_j, α_j) for $j \in \{1, \ldots, m\}$ are non-collinear, then the expected number of points on the convex hull grows as $O(\log m)$ [2]. Equivalently, there are $O(\log m)$ expected active nested constraints in the solution.

Note that a generalization of the previous reasoning is necessary to fully explain the results of our experiments since we considered $d_i \neq \infty$. Assuming that the same result holds in this more general case, then the amortized complexity of some methods such as [34] on randomly generated instances may be significantly better than the worst case. Indeed, this method iterates on the number of active constraints in an outer loop. The number of active constraints has no impact on the complexity and CPU time of the proposed method, but further pruning techniques may be investigated to eliminate constraints on the fly. Finally, the graphical approach used in this analysis leads to a strongly polynomial algorithm in $O(n + m \log m)$ for an interesting class of problems, and is worth further investigation on its own.

8. Conclusions. A dual-inspired approach has been introduced for NESTED resource allocation problems. The method solves NESTED as a hierarchy of simple resource allocation problems. The best known complexity of $O(n \log n \log \frac{B}{n})$ is attained for problems with as many nested constraints as variables, and a new best-known complexity of $O(n \log m \log \frac{B}{n})$ is achieved for problems with n variables and $\log m = o(\log n)$ nested constraints. Our computational experiments highlight significant CPU

time gains in comparison to other state-of-the-art methods on a wide range of problem instances with up to one million tasks.

The proposed algorithm relies on different principles than the previous state-of-the-art scaled greedy method. As such, it is not bound to the same methodological limitations and may be generalized to some problem settings with non-polymatroidal constraints, e.g., allocation problems with nested upper and lower constraints, which are also related to various key applications. Further pruning techniques exploiting the reduced number of active nested constraints can be designed and the geometric approach of Section 7 can be further investigated, aiming for generalization and an increased understanding of its scope of application. Finally, promising research perspectives relate to the extension of these techniques for various application fields, such as telecommunications and image processing, which can require to solve huge problems with similar formulations.

Appendix. To simplify the exposition, we assumed that $a_i \leq a_{i+1} \leq a_i + \sum_{k=s[i-1]+1}^{s[i]} d_k$ for $i \in \{1, \ldots, m\}$. If these conditions are not satisfied, then either the parameters a_i can be decreased to obtain an equivalent problem which fulfills them, or the problem can be declared to be infeasible. This transformation, described in Algorithm 3, takes $O(n)$ elementary operations.

Algorithm 3: Problem transformation and feasibility check

$a_0 \leftarrow 0$; $a_m \leftarrow B$
for $i = m - 1$ down to 1 **do**
  $a_i \leftarrow \min\{a_{i+1}, a_i\}$
for $i = 1$ to $m - 1$ **do**
  $a_i \leftarrow \min\{a_{i-1} + \sum_{k=s[i-1]+1}^{s[i]} d_k, a_i\}$
if $a_{m-1} + d_{m-1} < B$ **then**
  return Infeasible

References.

[1] E.D. Andersen, C. Roos, and T. Terlaky, *On implementing a primal-dual interior-point method for conic quadratic optimization*, Mathematical Programming, 95 (2003), pp. 249–277.
[2] G. Baxter, *A Combinatorial Lemma for Complex Numbers*, The Annals of Mathematical Statistics, 32 (1961), pp. 901–904.
[3] T. Bektas and G. Laporte, *The pollution-routing problem*, Transportation Research Part B: Methodological, 45 (2011), pp. 1232–1250.
[4] R.E. Bellman and S.E. Dreyfus, *Applied dynamic programming*, Princeton University Press, Princeton, NJ, 1962.
[5] R. Bellman, I. Glicksberg, and O. Gross, *The theory of dynamic programming as applied to a smoothing problem*, Journal of the Society for Industrial and Applied Mathematics, 2 (1954), pp. 82–88.
[6] E.B. Berman, *Resource allocation in a PERT network under continuous activity time-cost functions*, Management Science, 10 (1964), pp. 734–745.
[7] P. Brucker, *An O(n) algorithm for quadratic knapsack problems*, Operations Research Letters, 3 (1984), pp. 163–166.
[8] T.C.E. Cheng, A. Janiak, and M.Y. Kovalyov, *Bicriterion single machine scheduling with resource dependent processing times*, SIAM Journal on Optimization, 8 (1998), pp. 617–630.
[9] R. Deltheil, *Sur la théorie des probabilités géométriques*, PhD thesis, 1920.

[10] M.E. Dyer and J. Walker, *An algorithm for a separable integer programming problem with cumulatively bounded variables*, Discrete applied mathematics, 16 (1987), pp. 135–149.
[11] A. Federgruen and H. Groenevelt, *The greedy procedure for resource allocation problems: Necessary and sufficient conditions for optimality*, Operations Research, 34 (1986), pp. 909–918.
[12] S. Foldes and F. Soumis, *PERT and crashing revisited: Mathematical generalizations*, European Journal of Operational Research, 64 (1993), pp. 286–294.
[13] G.N. Frederickson and D.B. Johnson, *The complexity of selection and ranking in $X + Y$ and matrices with sorted columns*, Journal of Computer and System Sciences, 24 (1982), pp. 197–208.
[14] D.R. Fulkerson, *A network flow computation for project cost curves*, Management science, 7 (1961), pp. 167–178.
[15] H.H. Gabow and R.E. Tarjan, *A linear-time algorithm for a special case of disjoint set union*, Journal of Computer and System Sciences, 30 (1985), pp. 209–221.
[16] F. Hanssmann, *Determination of optimal capacities of service for facilities with a linear measure of inefficiency*, Operations Research, 5 (1957), pp. 713–717.
[17] H. Hashimoto, T. Ibaraki, S. Imahori, and M. Yagiura, *The vehicle routing problem with flexible time windows and traveling times*, Discrete Applied Mathematics, 154 (2006), pp. 2271–2290.
[18] D.S. Hochbaum, *Lower and upper bounds for the allocation problem and other nonlinear optimization problems*, Mathematics of Operations Research, 19 (1994), pp. 390–409.
[19] D.S. Hochbaum and S.-P. Hong, *About strongly polynomial time algorithms for quadratic optimization over submodular constraints*, Mathematical Programming, 69 (1995), pp. 269–309.
[20] D.S. Hochbaum and J.G. Shanthikumar, *Convex separable optimization is not much harder than linear optimization*, Journal of the ACM (JACM), 37 (1990), pp. 843–862.
[21] L.M. Hvattum, I. Norstad, K. Fagerholt, and G. Laporte, *Analysis of an exact algorithm for the vessel speed optimization problem*, Networks, 62 (2013), pp. 132–135.
[22] T. Ibaraki and N. Katoh, *Resource allocation problems: algorithmic approaches*, MIT Press, Boston, MA, 1988.
[23] J.E. Kelley and M.R. Walker, *Critical-path planning and scheduling*, in Proceedings of Eastern joint Computer conference, New York, 1959, ACM Press, pp. 160–173.
[24] H. Lakshmanan and D.P. de Farias, *Decentralized resource allocation in dynamic networks of agents*, SIAM Journal on Optimization, 19 (2008), pp. 911–940.
[25] N. Maculan, C.P. Santiago, E.M. Macambira, and M.H.C. Jardim, *An O(n) algorithm for projecting a vector on the intersection of a hyperplane and a box in $\mathbb{R}^{n1,2}$* , Journal of optimization theory and applications, 117 (2003), pp. 553–574.
[26] S.N. Majumdar, A. Comtet, and J. Randon-Furling, *Random convex hulls and extreme value statistics*, Journal of Statistical Physics, 138 (2010), pp. 955–1009.
[27] D.G. Malcolm, J.H. Roseboom, C.E. Clark, and W. Fazar, *Application*

of a technique for research and development program evaluation, Operations Research, 7 (1959), pp. 646–669.

[28] F. Modigliani and F.E. Hohn, *Production planning over time and the nature of the expectation and planning horizon*, Econometrica, 23 (1955), pp. 46–66.

[29] R.D.C. Monteiro and I. Adler, *An extension of Karmarkar type algorithm to a class of convex separable programming problems with global linear rate of convergence*, Mathematics of Operations Research, 15 (1990), pp. 408–422.

[30] S. Moriguchi and A. Shioura, *On Hochbaum's Proximity-Scaling Algorithm for the General Resource Allocation Problem*, Mathematics of Operations Research, 29 (2004), pp. 394–397.

[31] A.S. Nemirovsky and D.B. Yudin, *Problem complexity and method efficiency in optimization*, Wiley, New York, 1983.

[32] I. Norstad, K. Fagerholt, and G. Laporte, *Tramp ship routing and scheduling with speed optimization*, Transportation Research Part C: Emerging Technologies, 19 (2011), pp. 853–865.

[33] A. Padakandla and R. Sundaresan, *Power minimization for CDMA under colored noise*, IEEE Transactions on Communications, 57 (2009), pp. 3103–3112.

[34] ——, *Separable convex optimization problems with linear ascending constraints*, SIAM Journal on Optimization, 20 (2009), pp. 1185–1204.

[35] M. Patriksson, *A survey on the continuous nonlinear resource allocation problem*, European Journal of Operational Research, 185 (2008), pp. 1–46.

[36] D.W. Pentico, *The assortment problem: A survey*, European Journal of Operational Research, 190 (2008), pp. 295–309.

[37] D.R. Robinson, *A Dynamic Programming Solution to Cost-Time Tradeoff for CPM*, Management Science, 22 (1975), pp. 158–166.

[38] D. Ronen, *The effect of oil price on the optimal speed of ships*, Journal of the Operational Research Society, 33 (1982), pp. 1035–1040.

[39] W. Sadowski, *A few remarks on the assortment problem*, Management Science, 6 (1959), pp. 13–24.

[40] K.S. Srikantan, *A problem in optimum allocation*, Operations Research, 11 (1963), pp. 265–273.

[41] F.B. Talbot, *Resource-Constrained Project Scheduling with Time-Resource Tradeoffs: The Nonpreemptive Case*, Management Science, 28 (1982), pp. 1197–1210.

[42] A. Tamir, *Efficient algorithms for a selection problem with nested constraints and its application to a production-sales planning model*, SIAM Journal on Control and Optimization, 18 (1980), pp. 282–287.

[43] R.E. Tarjan, *Efficiency of a good but not linear set union algorithm*, Journal of the ACM, 22 (1975), pp. 215–225.

[44] A.F. Veinott, *Production planning with convex costs: A parametric study*, Management Science, 10 (1964), pp. 441–460.

[45] Z. Wang, *On Solving Convex Optimization Problems with Linear Ascending Constraints*, Optimization Letters, 9 (2015), pp. 819–838.