Separable convex optimization with nested lower and upper constraints

Thibaut VIDAL

Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro
Rua Marquês de São Vicente, 225 - Gávea, Rio de Janeiro - RJ, 22451-900, Brazil
vidalt@inf.puc-rio.br

SBPO'17, Blumenau

Contents

Content of this presentation based on three joint works:

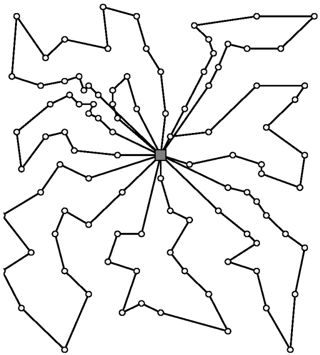
- T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins. **A unifying view on timing problems and algorithms**. Networks, 65(2), 102–128.
- T. Vidal, P. Jaillet, and N. Maculan, **A decomposition algorithm for nested resource allocation problems.** SIAM Journal on Optimization, 26(2), 1322–1340.
- T. Vidal, D. Gribel, and P. Jaillet, (2017). **Separable convex optimization with nested lower and upper constraints**. Submitted to Operations Research. Technical report PUC–Rio and MIT–LIDS. `https://arxiv.org/abs/1703.01484`.

Contents

Contents

Research context

- Capacitated vehicle routing problems (VRP)
  - **INPUT :** n customers, with locations and demand quantity. All-pair distances. Homogeneous fleet of m vehicles with capacity Q located at a central depot.
  - **OUTPUT :** Least-cost delivery routes (at most one route per vehicle) to service all customers.



- NP-Hard problem
- recent breakthrough in exact methods enable to solve problems of moderate size with up to 300-400 customers (Uchoa et al., 2013).
- A Scopus search “Vehicle Routing” for 2007-2011 returns 1258 publications, including 566 journal papers.
- Massive research on heuristics

Research context

- **Vehicle routing “attributes”:** Supplementary decisions, constraints and objectives which complement the classic VRP formulation.
  - modeling the specificities of application cases, customer requirements, network and vehicle specificities, operators abilities...
  - e.g., service time windows, multiple periods of planning, multiple depots and facilities, heterogeneous fleet, 2D-3D loading, time-dependent travel times...

- **Multi-Attribute Vehicle Routing Problems (MAVRP)**
  - Challenges: **VARIETY** of attributes
  - Challenges: **COMBINATION** of attributes
  - Plethora of attribute-specific methods in the literature, but highly problem specific
  - More **unified methods**, which can be extended to new problems without significant development, are necessary to answer the industrial needs in a timely manner.

Research Context

- General effort dedicated to better address rich vehicle routing problems involving many side constraints and attributes
- Observation : Many rich VRPs are hard because of their time features, e.g., (single, soft, or multiple) time windows, (time-dependent, flexible or stochastic) travel times, speed optimization, time-dependent costs, lunch breaks, HOS regulations...
- **Timing subproblems**:

  Given a fixed route, evaluate feasibility and cost w.r.t. time attributes

- Must be solved for all route and move evaluations

Research Context

- **Timing subproblems**:

GIVEN A FIXED ROUTE, EVALUATE FEASIBILITY AND COST W.R.T. TIME ATTRIBUTES

  - Review of timing problems and algorithms in [Vidal et. al, 2015, *Timing problems and algorithms: Time decisions for sequences of activities*. Networks, 65(2), 102–128].
  - More than 150 references, with efficient algorithms originally designed for other problems such as scheduling, PERT, resource allocation, isotone regression, telecommunications, machine learning...

- **Case 1) VRP with soft time windows**.
  Optimizing service dates for a given sequence of visits, in the presence of soft time windows $[e_i, l_i]$:

$$\min_{\mathbf{t}\geq\mathbf{0}} \ \alpha \sum_{i=1}^{n} \max\{e_i - t_i, 0\} + \beta \sum_{i=1}^{n} \max\{t_i - l_i, 0\} \tag{1.1}$$

$$\text{s.t.} \quad t_i + \delta_i \leq t_{i+1} \qquad 1 \leq i < n \tag{1.2}$$

$\Rightarrow$ Can be viewed as the optimization of a separable convex function over the order simplex:

$$\min \ f(\mathbf{x}) = \sum_{i=1}^{n} f_i(x_i) \tag{1.3}$$

$$\text{s.t.} \quad x_i \leq x_{i+1} \qquad i \in \{1, \dots, n-1\} \tag{1.4}$$

- **Case 1) VRP with soft time windows**.

  $\Rightarrow$ Can be viewed as the optimization of a separable convex function over the order simplex:

$$\min \quad f(\mathbf{x}) = \sum_{i=1}^{n} f_i(x_i) \tag{1.5}$$

$$\text{s.t.} \quad x_i \leq x_{i+1} \qquad i \in \{1, \ldots, n-1\} \tag{1.6}$$

- Interesting fact : 30 papers from various domains (routing, scheduling, PERT, isotonic regression) have been focused on this problem. All these papers can be reduced to three main algorithms (one primal approach, one dual, otherwise dynamic programming when for PL functions).

Research Context

- **Case 2) Vehicle speed optimization**.
Optimizing speed $\mathbf{v}$ over a fixed sequence of legs, making sure that service time-windows are respected, and f_i are convex functions

$$\min \; f(\mathbf{t},\mathbf{v}) = \sum_{i=2}^{n} \delta_{i-1,i} \; h_i(v_{i-1,i}) \tag{1.7}$$

$$\text{s.t.} \quad t_{i-1} + \frac{\delta_{i-1,i}}{v_{i-1,i}} \leq t_i \qquad i \in \{2,\ldots,n\} \tag{1.8}$$

$$a_i \leq t_i \leq b_i \qquad i \in \{1,\ldots,n\} \tag{1.9}$$

$$v_{min} \leq v_{i-1,i} \leq v_{max} \qquad i \in \{2,\ldots,n\}. \tag{1.10}$$

- Direct applications related to:
  - Ship speed optimization (Norstad et al., 2011; Hvattum et al., 2013)
  - Vehicle routing with flexible travel time or pollution routing (Hashimoto et al., 2006; Bektas and Laporte, 2011)

- **Case 2) Vehicle speed optimization**.
  After a quick reformulation:
  - With the change of variables $x_i = t_i - t_{i-1}$

$$\min \ f(\mathbf{x}) = \sum_{i=2}^{n} \delta_{i-1,i} \, g_i \left(\frac{\delta_{i-1,i}}{x_i} \right) \tag{1.11}$$

$$\text{s.t.} \ \ a_i \leq \sum_{k=1}^{i} x_k \leq b_i \qquad i \in \{1, \dots, n\} \tag{1.12}$$

$$\frac{\delta_{i-1,i}}{v_{max}} \leq x_i \qquad i \in \{2, \dots, n\}, \tag{1.13}$$

$$\text{with} \quad g_i(v) = \begin{cases} f_i(v_i^{\mathrm{OPT}}) & \text{if } v \leq v_i^{\mathrm{OPT}} \\ f_i(v) & \text{otherwise.} \end{cases} \tag{1.14}$$

Research Context

- **With simpler notations we obtain:**.

$$\min \; f(\mathbf{x}) = \sum_{i=1}^{n} f_i(x_i) \tag{1.15}$$

$$\text{s.t.} \quad a_i \leq \sum_{k=1}^{\sigma[i]} x_k \leq \; b_i \qquad i \in \{1, \ldots, m-1\} \tag{1.16}$$

$$\sum_{k=1}^{n} x_k = B \tag{1.17}$$

$$c_i \leq x_i \leq d_i \qquad i \in \{1, \ldots, n\}. \tag{1.18}$$

- "Resource Allocation Problem with Nested Constraints" (RAP–NC)
  - Special case where $a_i = -\infty$ called "NESTED"
  - Scope of this work : f_i convex & Lipschitz continuous but not necessarily differentiable or strictly convex.
  - For now, decision variables are continuous.

Research Context

- Without Equation (1.16), reduces to a simple Resource Allocation Problem:

$$\min \; f(\mathbf{x}) = \sum_{i=1}^{n} f_i(x_i) \tag{1.19}$$

$$\sum_{k=1}^{n} x_k = B \tag{1.20}$$

$$c_i \leq x_i \leq d_i \qquad\qquad i \in \{1, \ldots, n\}. \tag{1.21}$$

- Solvable in $\mathcal{O}(n)$ for linear or quadratic objectives, with either continuous or integer variables
- Solvable in $\mathcal{O}(n \log \frac{B}{n})$ for integer variables and convex objective.
- An ϵ-approximate solution of the continuous problem can be found in $\mathcal{O}(n \log \frac{B}{\epsilon})$ operations (to be explained later)

Research Context

- Ship speed optimization was our first motivation and application. The RAP–NC, however, is recurrent in a large variety of fields:

- **Lot Sizing** for example, with time-dependent production costs and inventory bounds:

$$
\begin{aligned}
\min \ f(\mathbf{x}, \mathbf{I}) = \sum_{i=1}^{n} p_i(x_i) + \sum_{i=1}^{n} \alpha_i I_i & & (1.22)\\
\text{s.t.} \quad I_i = I_{i-1} + x_i - d_i & \quad i \in \{2, \dots, n\} & (1.23)\\
I_0 = K & & (1.24)\\
0 \leq I_i \leq I_i^{\text{MAX}} & \quad i \in \{1, \dots, n\} & (1.25)\\
0 \leq x_i \leq x_i^{\text{MAX}} & \quad i \in \{1, \dots, n\}. & (1.26)
\end{aligned}
$$

Research Context

- **Lot Sizing** with time-dependent production costs and inventory bounds:
- Expressing the inventory variables as a function of the production quantities, using $I_i = K + \sum_{k=1}^{i}(x_k - d_k)$, we get

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

$$\begin{aligned} \text{s.t.} \quad & \sum_{k=1}^{i} d_k - K \leq \sum_{k=1}^{i} x_k \leq \sum_{k=1}^{i} d_k + I_i^{\text{MAX}} - K && i \in \{1, \ldots, n\} \\ & 0 \leq x_i \leq x_i^{\text{MAX}} && i \in \{1, \ldots, n\}. \end{aligned}$$

Research Context

- **Stratified Sampling:** Population of N units divided into subpopulations (*strata*) of $N_1, \ldots, N_n$ units s.t. $N_1 + \cdots + N_n = N$.
- Problem: determine the sample size $x_i \in [0, N_i]$ for each stratum, in order to estimate a characteristic of the population while ensuring a maximum variance level V and minimizing the total sampling cost.

$$\min \quad \sum_{i=1}^{n} c_i x_i \tag{1.27}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} \frac{N_i^2 \sigma_i^2}{N^2} \left(\frac{1}{x_i} - \frac{1}{N_i} \right) \leq V \tag{1.28}$$

$$0 \leq x_i \leq N_i \qquad i \in \{1, \ldots, n\}. \tag{1.29}$$

- In hierarchal sampling applications, may also need to bound the variance for subsets of stratums, as follows:

$$\sum_{i \in S_i} \frac{N_i^2 \sigma_i^2}{N^2} \left(\frac{1}{x_i} - \frac{1}{N_i} \right) \leq V_i, \qquad i \in \{1, \ldots, m\}, \tag{1.30}$$

Research Context

- **Stratified Sampling:** Population of N units divided into subpopulations (*strata*) of $N_1, \ldots, N_n$ units s.t. $N_1 + \cdots + N_n = N$.
- Problem: determine the sample size $x_i \in [0, N_i]$ for each stratum, in order to estimate a characteristic of the population while ensuring a maximum variance level V and minimizing the total sampling cost.

$$\min \quad \sum_{i=1}^{n} c_i x_i \tag{1.31}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} \frac{N_i^2 \sigma_i^2}{N^2} \left(\frac{1}{x_i} - \frac{1}{N_i} \right) \leq V \tag{1.32}$$

$$0 \leq x_i \leq N_i \qquad i \in \{1, \ldots, n\}. \tag{1.33}$$

- In hierarchal sampling applications, may also need to bound the variance for subsets of stratums, as follows:

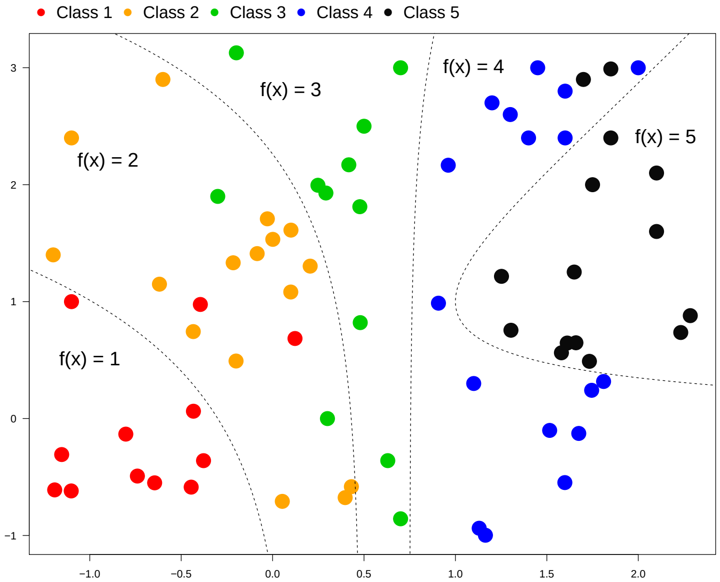
$$\sum_{i \in S_i} \frac{N_i^2 \sigma_i^2}{N^2} \left(\frac{1}{x_i} - \frac{1}{N_i} \right) \leq V_i, \qquad i \in \{1, \ldots, m\}, \tag{1.34}$$

Research Context

- **Machine Learning:** Support vector ordinal regression (SVOR) aims to find $r-1$ parallel hyperplanes so as to separate r ordered classes of samples in a kernel space. A dual formulation of this problem (Chu and Keerthi, 2007) can be formulated as follows:

$$
\begin{aligned}
\max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*,\boldsymbol{\mu}} \quad & \sum_{j=1}^{r}\sum_{i=1}^{n^j}(\alpha_i^j+\alpha_i^{*j})-\frac{1}{2}\sum_{j=1}^{r}\sum_{i=1}^{n^j}\sum_{j'=1}^{r}\sum_{i'=1}^{n^{j'}}(\alpha_i^{*j}-\alpha_i^j)(\alpha_{i'}^{*j'}-\alpha_{i'}^{j'})\mathcal{K}(x_i^j,x_{i'}^{j'}) \\
\text{s.t.} \quad & 0\le\alpha_i^j\le C \qquad j\in\{1,\ldots,r\}, i\in\{1,\ldots,n^j\} \\
& 0\le\alpha_i^{*j}\le C \qquad j\in\{1,\ldots,r-1\}, i\in\{1,\ldots,n^j\} \\
& \sum_{k=1}^{j}\left(\sum_{i=1}^{n^k}\alpha_i^k-\sum_{i=1}^{n^{k+1}}\alpha_i^{*k+1}\right)\ge 0 \qquad j\in\{1,\ldots,r-2\} \\
& \sum_{k=1}^{r-1}\left(\sum_{i=1}^{n^k}\alpha_i^k-\sum_{i=1}^{n^{k+1}}\alpha_i^{*k+1}\right)=0.
\end{aligned}
$$

Research Context



Research Context

- **Portfolio Optimization:** Mean-variance portfolio optimization (MVO) model of Markowitz (1952).
- In a simple form, maximize expected return while minimizing a risk measure such as the variance of the return. Can be formulated as:

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

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

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

  - x_i variables model the investments in different assets
  - μ_i is the expected return of asset i
  - σ_{ij} the covariance between asset i and j.

Research Context

- **Portfolio Optimization:** Mean-variance portfolio optimization (MVO) model of Markowitz (1952).
- In a simple form, maximize expected return while minimizing a risk measure such as the variance of the return. Can be formulated as:

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

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

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

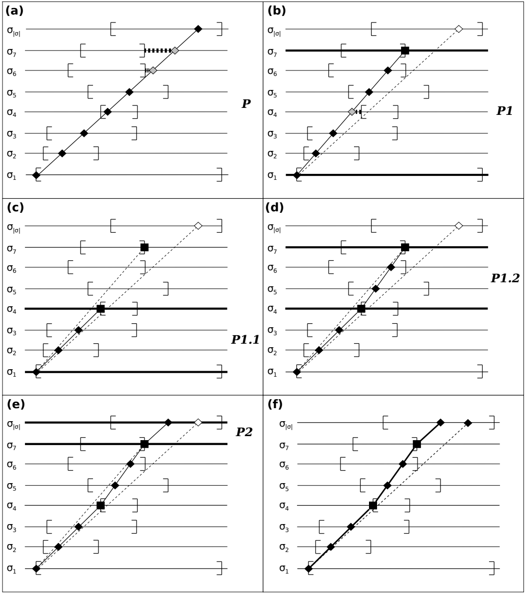
- But additional constraints can be considered
  - *Class constraints* limit the investment amounts for certain classes of assets or sector
  - *Fixed transaction costs, minimum transaction levels, and cardinality constraints...*

Contents

Existing algorithms – VRP or ship routing literature

- Recursive smoothing algorithm (Norstad et al., 2011; Hvattum et al., 2013)
  - **Applicable only when the cost/speed functions do not depend on the arc**
  - This case is strongly polynomial (which even never needs to evaluate the objective function)
  - Complexity : $O(n^2)$

Image from R. Kramer, A. Subramanian, T. Vidal, and L. A. F. Cabral. *A matheuristic approach for the Pollution-Routing Problem.* 2014. arXiv: 1404.4895v1



- This approach is closely related to the concept of *string method* (Dantzig 1971 and other earlier contributions)

(23)

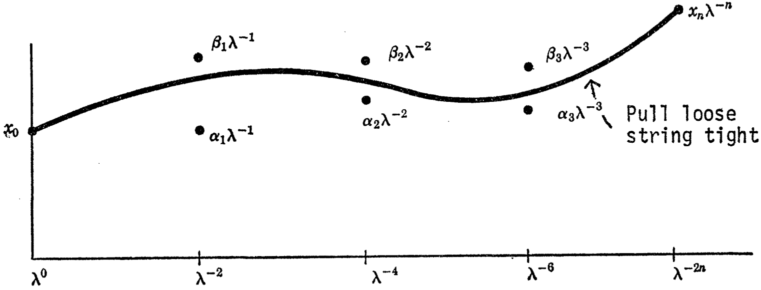


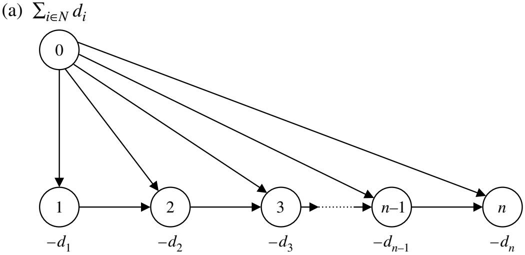
Image from G. B. Dantzig. A control problem of Bellman. Management Science. 17(9), pp. 542–546, 1971.

Existing algorithms – VRP & Lot Sizing literature

- **Dynamic programming approaches** for the case of **piecewise linear** functions (e.g., Hashimoto et al., 2006)
- Compute recursively the functions $F_i(b)$ which evaluate the minimum cost to execute the i first activities $(x_1, \ldots, x_i)$ with a resource consumption of b.

Existing algorithms – Lot Sizing literature

- **Flow algorithms** for the linear case, model the RAP–NC as the following min-cost flow problem:
  - Ahuja, R. K., & Hochbaum, D. S. (2008). *Technical note – Solving linear cost dynamic lot-sizing problems in O(n log n) time.* Operations Research, 56(1), 255–261.



  - Specialized *dynamic tree* structures allow to attain a complexity of $\mathcal{O}(n \log n)$. But very complex to implement.

Existing algorithms – NESTED

- **Dual-inspired methods**. Rely on the fact that the continuous resource allocation problem (RAP) can be solved by finding the zero of a single (Lagrangian) equation.
- Iteratively solving this equation and adjusting violated nested constraints.
  - Padakandla and Sundaresan (2009): complexity of $O(n^2 \Phi_{\text{RAP}}(n, B))$
  - Wang (2015): complexity of $O(n^2 \log n + n\Phi_{\text{RAP}}(n, B))$
  - where $\Phi_{\text{RAP}}(n, B)$ is the complexity of solving one RAP.

Existing algorithms – NESTED

- A greedy method with scaling for NESTED with integer variables (Hochbaum, 1994)
  - **Greedy** algorithms iteratively consider all feasible increments of one resource, and select the least-cost one.
  - **Convergence guarantee** (Federgruen and Groenevelt, 1986) to the optimum of the integer RAP

- **Scaling**.
  - An initial problem is solved with large increments
  - 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 to be corrected.
  - Complexity of $O(n \log n \log \frac{B}{n})$ for NESTED with integer variables

Wrap-up

Q: How can we solve efficiently the RAP–NC ?

A: If the objective is linear, apply the flow-based algorithm of Ahuja and Hochbaum (2008) in $\mathcal{O}(n \log n)$.

Q: But what if I have a general convex objective ?

A: Apply general-purpose convex optimization solvers, such as MOSEK or CVX

Q: But what if my problem is large ($n \geq 5,000$) or a fast answer is needed ?

A: ...

Wrap-up

Q: How can we solve efficiently the RAP–NC ?

A: If the objective is linear, apply the flow-based algorithm of Ahuja and Hochbaum (2008) in $\mathcal{O}(n \log n)$.

Q: But what if I have a general convex objective ?

A: Apply general-purpose convex optimization solvers, such as MOSEK or CVX

Q: But what if my problem is large ($n \geq 5,000$) or a fast answer is needed ?

A: ...

Wrap-up

Q: How can we solve efficiently the RAP–NC ?

A: If the objective is linear, apply the flow-based algorithm of Ahuja and Hochbaum (2008) in $\mathcal{O}(n \log n)$.

Q: But what if I have a general convex objective ?

A: Apply general-purpose convex optimization solvers, such as MOSEK or CVX

Q: But what if my problem is large ($n \geq 5,000$) or a fast answer is needed ?

A: ...

Wrap-up

Q: How can we solve efficiently the RAP–NC ?

A: If the objective is linear, apply the flow-based algorithm of Ahuja and Hochbaum (2008) in $\mathcal{O}(n \log n)$.

Q: But what if I have a general convex objective ?

A: Apply general-purpose convex optimization solvers, such as MOSEK or CVX

Q: But what if my problem is large ($n \geq 5,000$) or a fast answer is needed ?

A: ...

Wrap-up

Q: How can we solve efficiently the RAP–NC ?

A: If the objective is linear, apply the flow-based algorithm of Ahuja and Hochbaum (2008) in $\mathcal{O}(n \log n)$.

Q: But what if I have a general convex objective ?

A: Apply general-purpose convex optimization solvers, such as MOSEK or CVX

Q: But what if my problem is large ($n \geq 5,000$) or a fast answer is needed ?

A: ...

Wrap-up

Q: How can we solve efficiently the RAP–NC ?

A: If the objective is linear, apply the flow-based algorithm of Ahuja and Hochbaum (2008) in $\mathcal{O}(n \log n)$.

Q: But what if I have a general convex objective ?

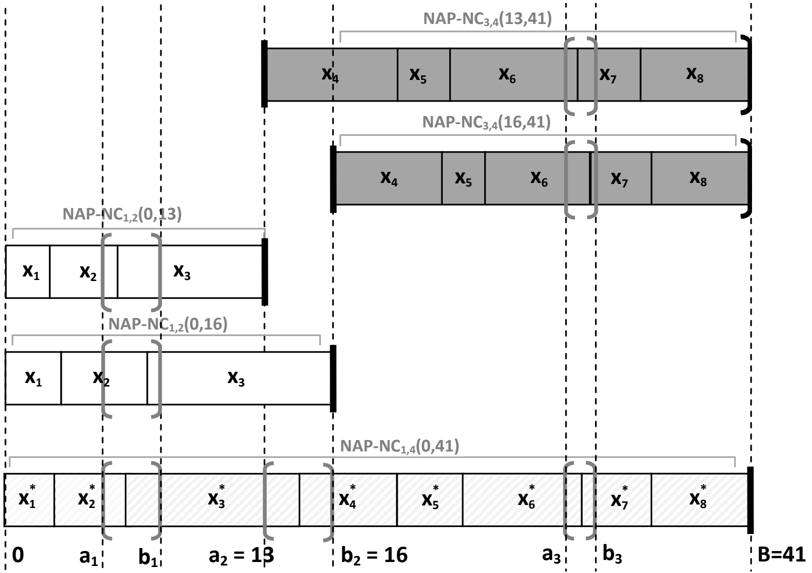
A: Apply general-purpose convex optimization solvers, such as MOSEK or CVX

Q: But what if my problem is large ($n \geq 5,000$) or a fast answer is needed ?

A: ...

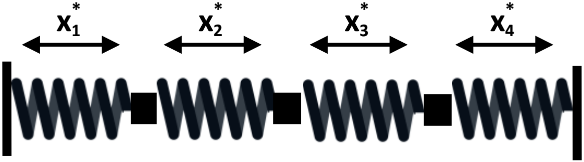
Contents

Proposed Algorithm – on an example



Proposed Algorithm – on an example

Think of the problem as a physical system made of springs:



Proposed Algorithm

- Divide the problem is easy. But how to exploit the information given by the subproblems to solve each iteration ? The answer comes from this:

Theorem (Monotonicity)

Consider three bounds $R^{\downarrow} \leq R \leq R^{\uparrow}$. If $\mathbf{x}^{\downarrow}$ is an optimal solution of $RAP\text{–}NC_{v,w}(L, R^{\downarrow})$ and $\mathbf{x}^{\uparrow}$ is an optimal solution of $RAP\text{–}NC_{v,w}(L, R^{\uparrow})$ such that $\mathbf{x}^{\downarrow} \leq \mathbf{x}^{\uparrow}$, then there exists an optimal solution $\mathbf{x}^{*}$ of $RAP\text{–}NC_{v,w}(L, R)$ such that $\mathbf{x}^{\downarrow} \leq \mathbf{x}^{*} \leq \mathbf{x}^{\uparrow}$.

Proof: Vidal, Gribel, & Jaillet, P. (2017). *Separable convex optimization with nested lower and upper constraints.* ArXiv report: https://arxiv.org/abs/1703.01484].

Proposed Algorithm

- This theorem allows to generate valid bounds (optimality cuts) on the variables, based on the information of the subproblems.
- BUT, these new inequalities have an important property, they are *stronger* than the nested constraints of the problem, i.e., if they are satisfied, then the nested constraints are satisfied:

$$x_k^{La} \leq x_k \leq x_k^{Lb} \text{ for } k \in \{\sigma[v-1]+1, \ldots, \sigma[u]\} \text{ and } i \in \{v, \ldots, u\}$$

$$\Rightarrow \sum_{k=\sigma[v-1]+1}^{\sigma[i]} x_k^{La} \leq \sum_{k=\sigma[v-1]+1}^{\sigma[i]} x_k \leq \sum_{k=\sigma[v-1]+1}^{\sigma[i]} x_k^{Lb}$$

$$\Rightarrow \bar{a}_i \leq \sum_{k=\sigma[v-1]+1}^{\sigma[i]} x_k \leq \bar{b}_i$$

Proposed Algorithm

- This theorem allows to generate valid bounds (optimality cuts) on the variables, based on the information of the subproblems.
- BUT, these new inequalities have an important property, they are *stronger* than the nested constraints of the problem, i.e., if they are satisfied, then the nested constraints are satisfied:

$$x_k^{La} \leq x_k \leq x_k^{Lb} \text{ for } k \in \{\sigma[v-1]+1, \dots, \sigma[u]\} \text{ and } i \in \{v, \dots, u\}$$

$$\Rightarrow \sum_{k=\sigma[v-1]+1}^{\sigma[i]} x_k^{La} \leq \sum_{k=\sigma[v-1]+1}^{\sigma[i]} x_k \leq \sum_{k=\sigma[v-1]+1}^{\sigma[i]} x_k^{Lb}$$

$$\Rightarrow \bar{a}_i \leq \sum_{k=\sigma[v-1]+1}^{\sigma[i]} x_k \leq \bar{b}_i$$

Proposed Algorithm

- Then, we can simply eliminate the nested constraints from the model and keep these new bounds on the variables, reducing each RAP–NC subproblem into a RAP.
- With this transformation, each level of the recursion can be solved with any classical RAP algorithm.

Proposed Algorithm

- Then, we can simply eliminate the nested constraints from the model and keep these new bounds on the variables, reducing each RAP–NC subproblem into a RAP.
- With this transformation, each level of the recursion can be solved with any classical RAP algorithm.

Proposed Algorithm – pseudo code

if $v = w$ **then**

$\quad (x^{aa}_{\sigma[v-1]+1}, \ldots, x^{aa}_{\sigma[v]}) \leftarrow \textsc{Rap}_{v,v}(a_{v-1}, a_w, -\infty, \infty)$;

$\quad (x^{ab}_{\sigma[v-1]+1}, \ldots, x^{ab}_{\sigma[v]}) \leftarrow \textsc{Rap}_{v,v}(a_{v-1}, b_w, -\infty, \infty)$;

$\quad (x^{ba}_{\sigma[v-1]+1}, \ldots, x^{ba}_{\sigma[v]}) \leftarrow \textsc{Rap}_{v,v}(b_{v-1}, a_w, -\infty, \infty)$;

$\quad (x^{bb}_{\sigma[v-1]+1}, \ldots, x^{bb}_{\sigma[v]}) \leftarrow \textsc{Rap}_{v,v}(b_{v-1}, b_w, -\infty, \infty)$;

else

$\quad u \leftarrow \lfloor \frac{v+w}{2} \rfloor$;

$\quad \mathrm{MDA}(v, u)$;

$\quad \mathrm{MDA}(u+1, w)$;

$\quad$ **for** $(L, R) \in \{(a,a), (a,b), (b,a), (b,b)\}$ **do**

$\qquad$ **for** $i = \sigma[v-1]+1$ to $\sigma[u]$ **do** $[\bar{c}_i, \bar{d}_i] \leftarrow [x^{La}_i, x^{Lb}_i]$;

$\qquad$ **for** $i = \sigma[u]+1$ to $\sigma[w]$ **do** $[\bar{c}_i, \bar{d}_i] \leftarrow [x^{bR}_i, x^{aR}_i]$;

$\qquad (x^{LR}_{\sigma[v-1]+1}, \ldots, x^{LR}_{\sigma[w]}) \leftarrow \textsc{Rap}_{v,w}(L, R, \bar{\mathbf{c}}, \bar{\mathbf{d}})$;

Proposed Algorithm – pseudo code

- **Q**: Does-it resolve the general convex case ?
- **A**: No, optimal solutions can be irrational (e.g., $\min f(x) = x^3 - 6x, x \geq 0$). What means "solving a subproblem" when we cannot even represent a solution?

- **Q**: Then, we cannot even represent the final solution of our problem in a bit-size computational model, it is ill defined...
- **A**: Indeed, this is why we do not require to solve *to optimality* a general convex problem. Instead, search for an ϵ-approximate solution, guaranteed to be located in the solution space no further than ϵ from an optimal solution.

- **Q**: Then, how can we control the imprecision of the algorithm at each layer of the recursion?
- **A**: This is not easy. We will give better ways than trying to work-around with numerical imprecisions in the method.

Proposed Algorithm – pseudo code

- **Q**: Does-it resolve the general convex case ?
- **A**: No, optimal solutions can be irrational (e.g., $\min f(x) = x^3 - 6x, x \geq 0$). What means "solving a subproblem" when we cannot even represent a solution?

- **Q**: Then, we cannot even represent the final solution of our problem in a bit-size computational model, it is ill defined...
- **A**: Indeed, this is why we do not require to solve *to optimality* a general convex problem. Instead, search for an ϵ-approximate solution, guaranteed to be located in the solution space no further than ϵ from an optimal solution.

- **Q**: Then, how can we control the imprecision of the algorithm at each layer of the recursion?
- **A**: This is not easy. We will give better ways than trying to work-around with numerical imprecisions in the method.

Proposed Algorithm – pseudo code

- **Q**: Does-it resolve the general convex case ?
- **A**: No, optimal solutions can be irrational (e.g., $\min f(x) = x^3 - 6x, x \geq 0$). What means "solving a subproblem" when we cannot even represent a solution?

- **Q**: Then, we cannot even represent the final solution of our problem in a bit-size computational model, it is ill defined...
- **A**: Indeed, this is why we do not require to solve *to optimality* a general convex problem. Instead, search for an ϵ-approximate solution, guaranteed to be located in the solution space no further than ϵ from an optimal solution.

- **Q**: Then, how can we control the imprecision of the algorithm at each layer of the recursion?
- **A**: This is not easy. We will give better ways than trying to work-around with numerical imprecisions in the method.

Proposed Algorithm – pseudo code

- **Q**: Does-it resolve the general convex case ?
- **A**: No, optimal solutions can be irrational (e.g., $\min f(x) = x^3 - 6x, x \geq 0$). What means "solving a subproblem" when we cannot even represent a solution?

- **Q**: Then, we cannot even represent the final solution of our problem in a bit-size computational model, it is ill defined...
- **A**: Indeed, this is why we do not require to solve *to optimality* a general convex problem. Instead, search for an ϵ-approximate solution, guaranteed to be located in the solution space no further than ϵ from an optimal solution.

- **Q**: Then, how can we control the imprecision of the algorithm at each layer of the recursion?
- **A**: This is not easy. We will give better ways than trying to work-around with numerical imprecisions in the method.

Proposed Algorithm – pseudo code

- **Q**: Does-it resolve the general convex case ?
- **A**: No, optimal solutions can be irrational (e.g., $\min f(x) = x^3 - 6x, x \geq 0$). What means "solving a subproblem" when we cannot even represent a solution?

- **Q**: Then, we cannot even represent the final solution of our problem in a bit-size computational model, it is ill defined...
- **A**: Indeed, this is why we do not require to solve *to optimality* a general convex problem. Instead, search for an ϵ-approximate solution, guaranteed to be located in the solution space no further than ϵ from an optimal solution.

- **Q**: Then, how can we control the imprecision of the algorithm at each layer of the recursion?
- **A**: This is not easy. We will give better ways than trying to work-around with numerical imprecisions in the method.

Proposed Algorithm – pseudo code

- **Q**: Does-it resolve the general convex case ?
- **A**: No, optimal solutions can be irrational (e.g., $\min f(x) = x^3 - 6x, x \geq 0$). What means "solving a subproblem" when we cannot even represent a solution?

- **Q**: Then, we cannot even represent the final solution of our problem in a bit-size computational model, it is ill defined...
- **A**: Indeed, this is why we do not require to solve *to optimality* a general convex problem. Instead, search for an ϵ-approximate solution, guaranteed to be located in the solution space no further than ϵ from an optimal solution.

- **Q**: Then, how can we control the imprecision of the algorithm at each layer of the recursion?
- **A**: This is not easy. We will give better ways than trying to work-around with numerical imprecisions in the method.

ϵ-approximate solutions

- Computational complexity of algorithms for general non-linear optimization problems $\Rightarrow$ an infinite output size may be needed due to real optimal solutions.

- To circumvent this issue
  - Existence of an oracle which returns the value of $f_i(x)$ in $O(1)$
  - Approximate notion of optimality (Hochbaum and Shanthikumar, 1990):

a continuous solution $\mathbf{x}^{(\epsilon)}$ is ϵ-accurate iff there exists an optimal solution $\mathbf{x}^*$ such that $||(\mathbf{x}^{(\epsilon)} - \mathbf{x}^*)||_\infty \leq \epsilon$.

  - Accuracy is defined in the solution space, in contrast with some other approximation approaches which considered objective space (Nemirovsky and Yudin, 1983).

Integer Variables and Proximity Theorem

- We will consider the integer problem, and use a proximity property between optimal continuous and integer solutions.

Theorem (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*

$$|x_i - x_i^*| < n - 1, \text{ for } i \in \{1, \ldots, n\}. \tag{3.1}$$

Special case of: Moriguchi, S., Shioura, A., & Tsuchimura, N. (2011). *M-convex function minimization by continuous relaxation approach: Proximity theorem and algorithm.* SIAM Journal on Optimization, 21(3), 633–668.

Integer Variables and Proximity Theorem

- **Q**: This allows to solve problems with integer variables now, but how this can help to find an ϵ-approximate solution for continuous problems ?
- **A**: To solve a continuous problem, simply transform the continuous problem into an integer problem where all parameters (a_i, b_i, c_i, d_i) have been scaled by a factor $\lceil n/\epsilon \rceil$, solve this problem (exactly, an integer solution is always representable) and transform back the solution. The proximity theorem guarantees that the solution is within the required precision.

Integer Variables and Proximity Theorem

- **Q**: This allows to solve problems with integer variables now, but how this can help to find an ϵ-approximate solution for continuous problems ?
- **A**: To solve a continuous problem, simply transform the continuous problem into an integer problem where all parameters (a_i, b_i, c_i, d_i) have been scaled by a factor $\lceil n/\epsilon \rceil$, solve this problem (exactly, an integer solution is always representable) and transform back the solution. The proximity theorem guarantees that the solution is within the required precision.

Computational Complexity

- **Convex objective.** Using the algorithms of Frederickson and Johnson (1982) or Hochbaum (1994) for the RAP subproblems $\Rightarrow$ complexity of $\mathcal{O}(n \log m \log B)$ for the RAP–NC with integer variables, and $\mathcal{O}(n \log m \log \frac{nB}{\epsilon})$ for an $\epsilon-$approximate solution of the continuous problem.

Computational Complexity

- **Quadratic objectives.** Using Ibaraki and Katoh (1988) in $\mathcal{O}(n)$ for the quadratic integer RAP, or Brucker (1984) in $\mathcal{O}(n)$ for the quadratic continuous RAP $\Rightarrow$ RAP–NC can be solved in $\mathcal{O}(n \log m)$, with either continuous or integer variables.
- This is the first strongly polynomial algorithm for the integer quadratic problem, responding positively to an open research question from Moriguchi et al. (2011): *"It is an open question whether there exist $\mathcal{O}(n \log n)$ algorithms for (Nest) with quadratic objective functions"*.

Computational Complexity

- **Linear objective.** Using a variant of median search in $\mathcal{O}(n)$ for the RAP $\Rightarrow$ RAP–NC can be solved in $\mathcal{O}(n \log m)$, with either continuous or integer variables.
- This is a slight improvement over the current network flow algorithm of Ahuja and Hochbaum (2008) in $\mathcal{O}(n \log n)$. It has the advantage of only using simple data structures, while the network flow algorithm relies on a *dynamic tree* (Tarjan, 1997; Tarjan and Werneck, 2009) or a *segment tree* (Bentley, 1977) with lazy propagation to keep track of capacity constraints.

Contents

Computational Experiments

- Three types of experiments:

- **Linear objective.** Comparison with the network flow algorithm of Ahuja and Hochbaum (2008)
- **Convex objective.** Comparison with MOSEK, a state-of-the-art convex optimization solver
- **Non-separable convex objective.** For the support vector ordinal regression problem (SVOR), using the RAP–NC as a subproblem in a projected gradient method.

Computational Experiments

- Tests on randomly-generated instances of the RAP–NC with a number of variables $n \in \{10, 20, 50, \dots, 10^6\}$, 10 instances per problem size
- For fine-grained analyses with the linear objective, 13×10 additional instances with $m = 100$ constraints and $n \in \{100, 200, 500, \dots, 10^6\}$ variables.
- Experiments with four classes of objectives: a linear objective $\sum_{i=1}^{n} p_i x_i$, and three convex objectives defined as:

$$
\begin{aligned}
\text{[F]} \qquad & f_i(x) = \frac{x^4}{4} + p_i x, \\
\text{[Crash]} \qquad & f_i(x) = k_i + \frac{p_i}{x}, \\
\text{and [Fuel]} \qquad & f_i(x) = p_i \times c_i \times \left(\frac{c_i}{x}\right)^3
\end{aligned}
$$

Experiments – Linear Objective

Table : Detailed CPU times for experiments with a linear objective

Variable m		CPU Time(s)	
n	m	FLOW	MDA
10	10	2.75×10^{-6}	4.78×10^{-6}
20	20	6.26×10^{-6}	1.02×10^{-5}
50	50	2.15×10^{-5}	2.85×10^{-5}
100	100	5.06×10^{-5}	5.89×10^{-5}
200	200	1.26×10^{-4}	1.26×10^{-4}
500	500	3.72×10^{-4}	3.36×10^{-4}
1000	1000	8.43×10^{-4}	7.57×10^{-4}
2000	2000	1.87×10^{-3}	1.74×10^{-3}
5000	5000	5.43×10^{-3}	5.20×10^{-3}
10000	10000	1.23×10^{-2}	1.12×10^{-2}
20000	20000	2.62×10^{-2}	3.21×10^{-2}
50000	50000	7.94×10^{-2}	1.05×10^{-1}
100000	100000	1.52×10^{-1}	2.26×10^{-1}
200000	200000	3.67×10^{-1}	4.86×10^{-1}
500000	500000	9.68×10^{-1}	1.37
1000000	1000000	1.99	2.98

Fixed m		CPU Time(s)	
n	m	FLOW	MDA
100	100	5.09×10^{-5}	5.95×10^{-5}
200	100	1.36×10^{-4}	1.26×10^{-4}
500	100	3.94×10^{-4}	2.86×10^{-4}
1000	100	9.07×10^{-4}	5.52×10^{-4}
2000	100	2.07×10^{-3}	1.14×10^{-3}
5000	100	6.16×10^{-3}	2.96×10^{-3}
10000	100	1.44×10^{-2}	6.26×10^{-3}
20000	100	3.17×10^{-2}	1.57×10^{-2}
50000	100	9.27×10^{-2}	5.26×10^{-2}
100000	100	2.04×10^{-1}	1.08×10^{-1}
200000	100	4.41×10^{-1}	2.36×10^{-1}
500000	100	1.20	7.19×10^{-1}
1000000	100	2.56	1.60

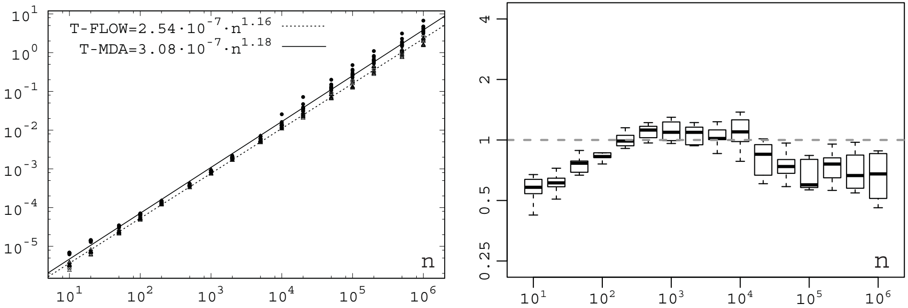


Figure : Varying $n \in \{10, \dots, 10^6\}$ and $m = n$. Left figure: CPU time of both methods as n and m grow. Right figure: Boxplots of the ratio $T_{\mathrm{FLOW}}/T_{\mathrm{MDA}}$.

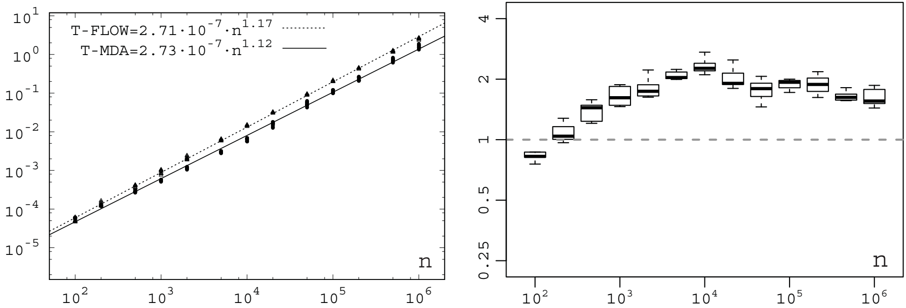


Figure : Linear Objective. Varying $n \in \{10, \ldots, 10^6\}$ and fixed $m = 100$. Left figure: CPU time of both methods as n grows. Right figure: Boxplots of the ratio $T_{\text{FLOW}}/T_{\text{MDA}}$.

Experiments – Convex Objective

Table : Detailed CPU-time for 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	5.45×10^{-3}	3.55×10^{-3}	6.21×10^{-3}	7.79×10^{-1}	7.56×10^{-1}	7.86×10^{-1}
1000	1000	1.27×10^{-2}	8.61×10^{-3}	1.43×10^{-2}	6.31	6.29	6.37
2000	2000	2.88×10^{-2}	1.87×10^{-2}	3.19×10^{-2}	8.57×10^{1}	9.38×10^{1}	9.05×10^{1}
5000	5000	9.27×10^{-2}	6.05×10^{-2}	9.86×10^{-2}	1.70×10^{3}	1.61×10^{3}	1.55×10^{3}
10000	10000	2.01×10^{-1}	1.34×10^{-1}	2.13×10^{-1}	—	—	—
20000	20000	4.69×10^{-1}	3.04×10^{-1}	4.82×10^{-1}	—	—	—
50000	50000	1.31	8.74×10^{-1}	1.33	—	—	—
100000	100000	3.12	2.02	3.07	—	—	—
200000	200000	6.68	4.58	6.61	—	—	—
500000	500000	1.98×10^{1}	1.35×10^{1}	1.91×10^{1}	—	—	—
1000000	1000000	4.54×10^{1}	3.10×10^{1}	4.30×10^{1}	—	—	—

Experiments – Convex Objective

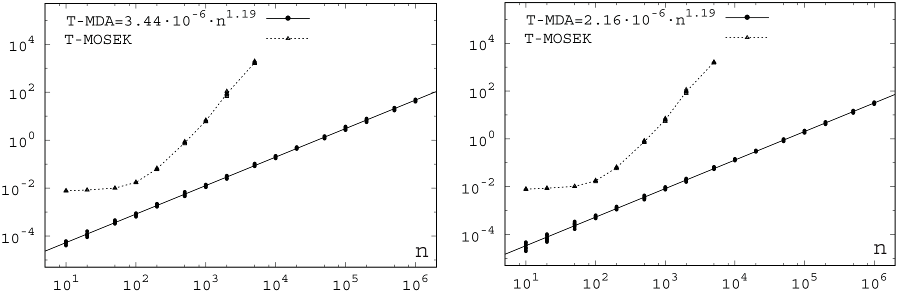


Figure : CPU time of MDA and MOSEK as n grows and $m = n$ for the objectives [F] and [Crash].

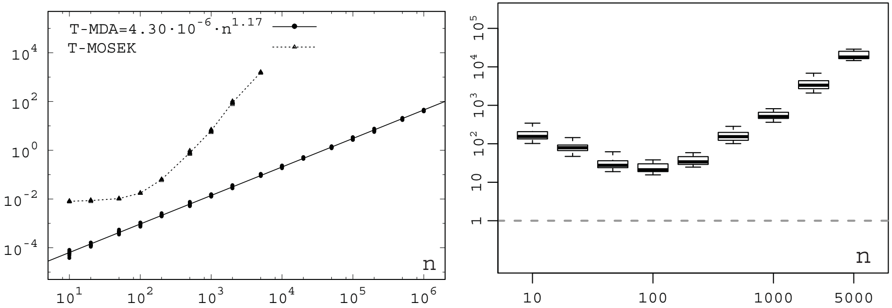


Figure : Left Figure: CPU time of MDA and MOSEK as n grows and $m = n$ for objective [Fuel]. Right Figure: Boxplots of the ratio $T_{\text{Mosek}}/T_{\text{MDA}}$.

Experiments – Non-Separable Convex Objective

- Last experimental analysis is concerned with the SVOREX model
- A non-separable convex optimization problem over a special case of the RAP–NC constraint polytope.

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

$$\begin{aligned}
\text{s.t.}\quad & 0\le\alpha_i^j\le C && j\in\{1,\dots,r\}, i\in\{1,\dots,n^j\}\\
& 0\le\alpha_i^{*j}\le C && j\in\{1,\dots,r-1\}, i\in\{1,\dots,n^j\}\\
& \sum_{k=1}^{j}\left(\sum_{i=1}^{n^k}\alpha_i^k-\sum_{i=1}^{n^{k+1}}\alpha_i^{*k+1}\right)\ge 0 && j\in\{1,\dots,r-2\}\\
& \sum_{k=1}^{r-1}\left(\sum_{i=1}^{n^k}\alpha_i^k-\sum_{i=1}^{n^{k+1}}\alpha_i^{*k+1}\right)=0.
\end{aligned}$$

Experiments – Non-Separable Convex Objective

- Current state-of-the-art algorithm for this problem, proposed by Chu and Keerthi (2007), based on a working-set decomposition.
- Iteratively, a set of variables is selected to be optimized over, while the others remain fixed.
- This approach leads to a (non-separable) restricted problem with fewer variables which can be solved to optimality.

Experiments – Non-Separable Convex Objective

- Chu and Keerthi (2007) use a *minimal working set* containing the two variables which most violates the KKT conditions
  - **Advantage:** Availability of analytical solutions for the restricted problems
  - **Drawback:** Large number of iterations until convergence

- Our RAP–NC algorithm can provide another meaningful option
  - **Generating larger working sets**, and solving the resulting reduced problems with the help of the RAP–NC algorithm
  - **Warning: the reduced problems are non-separable** ⇒ RAP–NC algorithm is used for the projection steps within a projected gradient descent procedure

Experiments – Non-Separable Convex Objective

- Chu and Keerthi (2007) use a *minimal working set* containing the two variables which most violates the KKT conditions
  - **Advantage:** Availability of analytical solutions for the restricted problems
  - **Drawback:** Large number of iterations until convergence

- Our RAP–NC algorithm can provide another meaningful option
  - **Generating larger working sets**, and solving the resulting reduced problems with the help of the RAP–NC algorithm
  - **Warning: the reduced problems are non-separable** ⇒ RAP–NC algorithm is used for the projection steps within a projected gradient descent procedure

Experiments – Non-Separable Convex Objective

- Chu and Keerthi (2007) use a *minimal working set* containing the two variables which most violates the KKT conditions
  - **Advantage:** Availability of analytical solutions for the restricted problems
  - **Drawback:** Large number of iterations until convergence

- Our RAP–NC algorithm can provide another meaningful option
  - **Generating larger working sets**, and solving the resulting reduced problems with the help of the RAP–NC algorithm
  - **Warning: the reduced problems are non-separable** $\Rightarrow$ RAP–NC algorithm is used for the projection steps within a projected gradient descent procedure

Experiments – Non-Separable Convex Objective

- Chu and Keerthi (2007) use a *minimal working set* containing the two variables which most violates the KKT conditions
  - **Advantage:** Availability of analytical solutions for the restricted problems
  - **Drawback:** Large number of iterations until convergence

- Our RAP–NC algorithm can provide another meaningful option
  - **Generating larger working sets**, and solving the resulting reduced problems with the help of the RAP–NC algorithm
  - **Warning: the reduced problems are non-separable** $\Rightarrow$ RAP–NC algorithm is used for the projection steps within a projected gradient descent procedure

Experiments – Non-Separable Convex Objective

$\boldsymbol{\alpha} = \boldsymbol{\alpha}^* = \mathbf{0}$; `// Initial Solution set to 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**

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

    `// Solve the projection subproblem as a RAP-NC`

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

Experiments – Non-Separable Convex Objective

- Experiments with working sets of size $n_{\text{WS}} \in \{2, 4, 6, 10\}$, a step size of $\gamma = 0.2$ and $n_{\text{GRAD}} = 20$ iterations for the projected gradient descent.
- Eight data sets from Chu and Keerthi (2007)

Experiments – Non-Separable Convex Objective

Table : SVOREX resolution – impact of the working-set size

Instance	N	D	Solution Variables s.t. $\alpha = 0$	$\alpha = C$	$\alpha \in]0, C[$	n_{ws}	I_{ws}	T(s)
Abalone	1000	8	39%	32%	29%	**2**	**118233**	**13.46**
						4	96673	21.51
						6	78433	26.34
						10	60605	35.46
Bank	3000	32	25%	0%	75%	2	139468	68.41
						4	52073	63.02
						6	**31452**	**45.22**
						10	21310	47.66
Boston	300	13	41%	0%	59%	2	7207	0.43
						4	**3697**	**0.40**
						6	2840	0.46
						10	2076	0.54
California	5000	8	51%	43%	6%	**2**	**250720**	**124.46**
						4	189289	185.79
						6	166879	245.08
						10	146170	360.52

Table : SVOREX resolution – impact of the working-set size

Instance	**N**	**D**	**Solution Variables s.t.** $\alpha = 0$	$\alpha = C$	$\alpha \in]0, C[$	$\mathbf{n}_{\mathrm{ws}}$	$\mathbf{I}_{\mathbf{ws}}$	**T(s)**
Census	6000	16	38%	4%	59%	**2**	**349894**	**242.11**
						4	206951	301.74
						6	180608	393.28
						10	155731	574.28
Computer	4000	21	64%	32%	4%	2	290207	168.94
						4	140270	161.45
						6	**98948**	**153.56**
						10	68616	193.10
Machine CPU	150	6	49%	9%	41%	2	28856	1.24
						4	**11534**	**0.86**
						6	8144	0.91
						10	6363	1.24
Pyrimidines	50	27	21%	0%	79%	2	935	0.035
						4	367	0.021
						6	**218**	**0.018**
						10	144	0.023

Contents

Conclusions and Perspectives

- RAP–NC: **wide range of applications** in production and transportation optimization, portfolio management, sampling optimization, telecommunications and machine learning.
- A new type of decomposition method, based on **monotonicity principles coupled with divide-and-conquer**
- **Complexity breakthroughs**, and first known strongly polynomial algorithm for the quadratic integer RAP–NC
- **Good practical performance**, and applications to ordinal regression problems for machine learning

Conclusions and Perspectives

- **Very different principles:** not based on classical greedy steps and scaling, or on flow propagation techniques.

- Key research questions $\Rightarrow$ How far this type of decomposition can be generalized
  - Other convex resource allocation problems where, e.g., the constraints follow a TREE of lower *and upper* constraints (Hochbaum, 1994)
  - Extended formulations involving the intersection of two or more RAP–NC type of constraint polytopes
  - ...

Thank you

THANK YOU FOR YOUR ATTENTION !

- T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins. **A unifying view on timing problems and algorithms**. Networks, 65(2), 102–128.
- T. Vidal, P. Jaillet, and N. Maculan, **A decomposition algorithm for nested resource allocation problems.** SIAM Journal on Optimization, 26(2), 1322–1340.
- T. Vidal, D. Gribel, and P. Jaillet, (2017). **Separable convex optimization with nested lower and upper constraints**. Submitted to Operations Research. Technical report PUC–Rio and MIT–LIDS. `https://arxiv.org/abs/1703.01484`.
- `http://w1.cirrelt.ca/~vidalt/`

Bibliography I

Ahuja, R.K., D.S. Hochbaum. 2008. Technical note – Solving linear cost dynamic lot-sizing problems in O(n log n) time. *Operations Research* **56**(1) 255–261.

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

Bentley, J.L. 1977. Solutions to Klee's rectangle problems. Tech. rep., Carnegie-Mellon University, Pittsburgh PA.

Brucker, P. 1984. An O(n) algorithm for quadratic knapsack problems. *Operations Research Letters* **3**(3) 163–166.

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

Dantzig, G.B. 1971. A control problem of Bellman. *Management Science* **17**(9) 542–546.

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

Frederickson, G.N., D.B. Johnson. 1982. The complexity of selection and ranking in X + Y and matrices with sorted columns. *Journal of Computer and System Sciences* **24**(2) 197–208.

Hashimoto, H., T. Ibaraki, S. Imahori, M. Yagiura. 2006. The vehicle routing problem with flexible time windows and traveling times. *Discrete Applied Mathematics* **154**(16) 2271–2290.

Bibliography II

Hochbaum, D.S. 1994. Lower and upper bounds for the allocation problem and other nonlinear optimization problems. *Mathematics of Operations Research* **19**(2) 390–409.

Hochbaum, D.S., J.G. Shanthikumar. 1990. Convex separable optimization is not much harder than linear optimization. *Journal of the ACM (JACM)* **37**(4) 843–862.

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

Ibaraki, T., N. Katoh. 1988. *Resource allocation problems: algorithmic approaches*. MIT Press, Boston, MA.

Markowitz, H. 1952. Portfolio selection. *The Journal of Finance* **7**(1) 77–91.

Moriguchi, S., A. Shioura, N. Tsuchimura. 2011. M-convex function minimization by continuous relaxation approach: Proximity theorem and algorithm. *SIAM Journal on Optimization* **21**(3) 633–668.

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

Norstad, I., K. Fagerholt, G. Laporte. 2011. Tramp ship routing and scheduling with speed optimization. *Transportation Research Part C: Emerging Technologies* **19**(5) 853–865.

Padakandla, A., R. Sundaresan. 2009. Power minimization for CDMA under colored noise. *IEEE Transactions on Communications* **57**(10) 3103–3112.

Tarjan, R.E. 1997. Dynamic trees as search trees via Euler tours, applied to the network simplex algorithm. *Mathematical Programming* **78**(2) 169–177.

Bibliography III

Tarjan, R.E., R.F. Werneck. 2009. Dynamic trees in practice. *Journal of Experimental Algorithmics* **14** 5–23.

Wang, Z. 2015. On Solving Convex Optimization Problems with Linear Ascending Constraints. *Optimization Letters* **9** 819–838.