Fast dynamic programming with application to storage planning

Robin Girard, Vincent Barbesant, Fiona Foucault, Georges Kariniotakis

Fast dynamic programming with application to storage planning

Robin Girard*, Vincent Barbesant**, Fiona Foucault*, Georges Kariniotakis*
*Center PERSEE of MINES ParisTech, robin.girard@mines-paristech.fr, **RTE

***Abstract*—In this paper we propose a new class of algorithms that allows to solve a class of optimization problems including that of finding an optimal storage policy. The proposed algorithm is fast with a quadratic time or even quasi linear time in some cases. The gain in term of complexity with regard to the use of a commercial linear solver is shown empirically. A freely available package in the R statistical software has been implemented and is presented here. Possible extension to more complex framework are presented. This paper ends with an example of application for the determination of the market revenue of a storage device in a local price market as a function of location, energy capacity and power limits.**

***Index Terms*—Dynamic programming, optimization, storage, planing, wind power forecast, local price market.**

I. Introduction

THE sustainability of electric power systems is an important milestone in our path towards decarbonization. This involves the large scale utilization of renewable energy sources with the most prominent contribution from wind and solar energy, as well as biomass, to gradually replace fossil fuels to produce (mainly electrical) energy. This is encouraged by policies e.g. goals have been set for member states of the European Union to reach a 20% share of renewable energy in its overall final energy consumption by 2020.

The actual claim is that the increased share of fluctuating renewable power generation in the electricity system will require a significant investment into storage and transport capacities. Among the different applications of storage which are considered in the literature, one can distinguish between those mitigating constraints at the local level (e.g. capacity or voltage constraint in the network), and those mitigating global issues. The use of storage as reserve belongs to this latter category e.g. to compensate for errors in the consumption or PV/wind power production forecasts as well as for the intra-hourly load following but more importantly to displace energy allowing to meet hourly energy demand at the best price. In the liberalized electricity market, this theoretically translates into a use of storage which becomes economically viable from buying the energy when it is cheap and selling it at a higher price.

The literature about the use of storage and its optimization is vast but algorithms which allow to take several applications into consideration while running fast enough to evaluate the benefit of storage for a year in a few millisecond (thus allowing to run it for thousands of configurations) are missing even with existing commercial software.

This work was partially funded by the ANR project WinPower.

In [1] the proposed procedure considers the mitigation of wind power forecast errors but relies on a loop over several disconnected decisions along the year. Each considered decision is a scheduling problem where the temporal horizon is relatively small (i.e. few hours to one or two days ahead). Dynamic programming philosophy has been used for a long time now in the storage optimization field because it allows the use of non-linear cost functions (see e.g. [2]). However, the general use of Bellman's idea traditionally necessitates a discretization of the state-space and results in solutions which are not computationally more efficient than linear commercial solvers in the case of linear problems [3]. As a result, only few temporal horizons are considered (in [1], [2], [4]) or weekly averages are used [3].

In this paper we propose an algorithm to solve a large class of linear or quadratic optimization problems including the one of storage scheduling when the temporal horizon is large (several thousands to millions). It finds an exact solution in a few milliseconds. The general algorithm together with the description of the optimization problem it solves are given in Section II. To our knowledge, this algorithm is new and relies on the use of ideas from existing algorithms [5] for the computation of a chain of infimum convolutions transform together with dynamic programming reasoning. We propose an implementation in an open source package available into the R software for statistical computing [6]. This implementation takes the form of a package which is documented and publicly available via the CRAN server hence promoting the idea of reproducible research. We propose several extensions which are described in Section III: the stochastic case, a case of non-linear constraints and the case of multivariate state space. The three extensions are implemented in the R package. We finally present in Section IV an application to the case of storage dimensioning and positioning for price balancing. This application is carried out in the context of a local price market using hourly price data and imbalance penalties data from the American PJM market. Section V draws conclusions and perspectives.

II. Storage use optimization and dynamical programming

In this section we describe the class of optimization problem that we will consider in the remainder of this article and we propose an algorithm to solve it. This class is introduced through a simple storage problem. This Section aims at providing general ideas and principles, whereas further extensions of the class and algorithm are presented in the next Section.

A. Problem

Let us introduce the class of optimization problem considered here :

$$\begin{array}{ll}\min & \sum_{i=1}^{n} C_i(x_i) \\ s.t. & \begin{cases} lb_i \leq x_i \leq ub_i & i = 1,..,n \\ lbC_i \leq \sum_{j=1}^{i} x_j \leq ubC_i & i = 1,..,n \end{cases}\end{array} \quad (1)$$

where lbC,$ulbC$, lb and ub are vectors of $\mathbb{R}^n$, and C_i ($i = 1 \ldots, n$) are cost functions. For now, no specific assumption is required on the cost functions. A specific instance of this problem appears when one wants to minimize the cost of a storage use under energy capacity constraint and power constraint. For example in the case when C_i is linear with slope π_i, x_i is the stored power at time i $\pi_i x_i$ is the cost. Negative cost (i.e. benefit) appears when energy is sold ($x_i < 0$) at the moment when the prices are higher. The first line of constraints are the power limits and the second set of constraints are energy capacity constraints. The general form given by Equation 1 may for instance allow to include storage round trip efficiency, network taxes (with piecewise linear C_i, one piece for $x_i < 0$ and one piece for $x_i > 0$), or the price sensitivity to the use of storage in the electricity market.

B. Dynamic programming and the general algorithm

1) Dynamic programming formulation : the recurrence equations: Dynamic programming was introduced in [7] as a methodology to transform a high dimensional optimization problem into that of finding an optimal policy which can take the form of a recurrence equation on the state of the recurrence system. We introduce the function $D_k(z)$, defined for all $k \in \mathbb{N}^*$:

$$D_k(z) = \min \left(\sum_{i=1}^{k} C_i(x_i) \right)$$
$$s.t. \begin{cases} x_i \in [lbP_i; ubP_i] & i = 1,..,k \\ \sum_{j=1}^{i} x_j \in [lbC_i; ubC_i] & i = 1,..,k-1 \\ \sum_{j=1}^{n} x_j = z \end{cases}$$

In our case, a classical dynamic programming reasoning (i.e. the so called Bellman principle) gives the recurrence equation for all $k = 2, \ldots, n$:

$$D_k(z) = \min \left(C_k(x) + D_{k-1}(z - x) \right) \quad (2)$$
$$s.t. \begin{cases} x \in [lbP_k; ubP_k] \\ z - x \in [lbC_{k-1}; ubC_{k-1}] \end{cases}$$

and when for all $k = 1, \ldots, n$ D_k have been computed, the state of the storage z_n at terminal state is the state that minimizes D_n while the state of the storage z_k^* at step $k < n$ is obtained through the following descending recurrence equation:

$$z_k^* = arg\min \left(C_{k+1}(x) + D_k(z_{k+1}^* - x) \right) \quad (3)$$
$$s.t. \begin{cases} x \in [lbP_{k+1}; ubP_{k+1}] \\ z_{k+1}^* - x \in [lbC_k; ubC_k] \end{cases}$$

2) Formulation of the algorithm with elementary operations: The operator *InfConv* (for Infimal convolution, also called *Epi-Sum*) appears in the Equation (3) :

$$(f \square g)(x) = \min_{y \in \mathbb{R}} \{f(x - y) + g(y)\}$$

This operator has a well known microeconomic interpretation (see e.g. [8]): if f_1 and f_2 are two cost functions associated to two production units, $(f_1 \square f_2)(x)$ represents the joint cost for a production level x when this production level is shared out among the different units in the most efficient possible way.

In order to integrate the two constraints of the problem, the functions have to be restricted to a smaller domain of definition. To carry out this operation, the *squeeze* operator is used :

$$f[a,b](x) = \begin{cases} f(x) & \text{if} \quad x \in [a,b] \\ +\infty & \text{otherwise} \end{cases}$$

Another simple operator is the *swap* operator :

$$(\oslash[f,y])(x) = f(y - x) \quad (4)$$

Thus, with these notations, the translation of the recurrence Equations (3) and (4) into an algorithm using only these operators is straightforward and gives:

Algorithm 1 Dynamic programming approach for solving problem 1 with only operators *squeeze*, *sum*, *swap* and *InfConv*

$D_1 = C_1[lbP_1, ubP_1]$
for $i = 2 \to n$ **do**
  $D_i(z) = (D_{i-1}[lbC_i, ubC_i]) \square (C_i[lbP_i, ubP_i])$
end for
$z_n^* = D_n[lbC_n, ubC_n]$
for $i = n - 1 \to 1$ **do**
  $f = \oslash \left[D_i[lbC_i, ubC_i], z_{i+1}^* \right] + C_{i+1}[lbP_{i+1}, ubP_{i+1}]$
  $z_i^* = \text{Argmin} f$
end for

3) The case of piecewise polynomial functions, a fast exact algorithm: The important point about Algorithm 1 is that when the cost functions $(C_i)_{i=1,\ldots,n}$ belong to an adapted function class $\mathcal{C}$ it is possible to show that :

- the algorithm is stable i.e. $\forall i \in \{1, \ldots, n\}$ $C_i \in \mathcal{C}$ implies $\forall i \in \{1, \ldots, n\}$ $D_i \in \mathcal{C}$.
- all elementary operations which are used may have a fast algorithm, i.e. an algorithm running in a linear or log-linear time.

For example, it is the case for piecewise polynomial convex functions. Indeed, for a convex function f one can define its Legendre transform f^* (also known as convex conjugate) by:

$$f^*(p) = \sup_{x \in \mathbb{R}} (px - f(x)), \forall p \in \mathbb{R}$$

It is well known that f^* is also convex, and satisfies :

$$(f^*)^* = f \tag{5}$$

thus defining a reversible transformation between a x-space and a p-space.

From these definitions, it is well known [9] that the operators $\square$ and $+$ are dual to each other through conjugation:

$$(f \square g)^* = f^* + g^* \tag{6}$$

We can transpose the operation in the first loop of Algorithm (II-B2) through conjugation to obtain:

$$D_n^* = C_n[lb_n, ub_n]^* + D_{n-1}[lbC_n, ubC_n]^*$$

In addition, if f is polynomial, its Legendre transform can be computed explicitly, and if it is a convex polynomial whose order is d at most, so will be its Legendre transform. The case of piecewise polynomials follows immediately by treating the function f piece by piece. This is possible here because of the convexity of f, as a discontinuity in the first order derivative results in a linear piece and a linear piece is transformed into a discontinuity in the first order. In particular, in the case of piecewise linear convex functions, the Legendre transform is merely an exchange of breakpoints and slopes. These results are well known and further details are found in [10].

The algorithmic complexity in the case when $\mathcal{C}$ is either the set of piecewise linear functions or the set of piecewise quadratic functions as well as the chosen implementation and further development are discussed in the extended version of this paper. The corresponding R package ConConPiWiFun is available on the CRAN server. We only show here the experimental complexity comparison of our tool with a numerical solver for linear programming.

4) Empirical complexity comparison with an interior point algorithm: In the case of problem 1 with linear cost function it is straightforward to use directly a state of the art linear programming tool using for instance interior points which are known to have a polynomial complexity but with an unknown degree. In the more general case, e.g. with piecewise linear cost functions, it is possible to recover a linear programming problem by adding more constraints and more variables to the initial problem, increasing the number of constraints. Practically, even in the first, simple case, the proposed procedure and its implementation are more efficient than those obtained using the up to date commercial solver CPLEX to solve directly the linear programming problem. The corresponding benchmark results are shown in Table I. The solved problem was the one defined by Equation 1 with random prices (slopes of the cost functions). Results are presented as a function of the temporal horizon n. Over a size of $n = 10000$ the memory was an issue with the linear programming requiring building a constraint matrix of size $n \times 2n$, and this resulted in non computed values (NC) in Table I.

TABLE I
EMPIRICAL SPEED/COMPLEXITY COMPARISON BETWEEN THE PROPOSED DYNAMIC PROGRAMMING METHOD AND THE USE OF A STATE OF THE ART LINEAR PROGRAMMING SOLVER (CPLEX).

n	100	1000	5000	10^4	10^5	10^6
dyn.prog. [ms]	0.34	3.06	28.8	61.5	649.2	6285
CPLEX [ms]	4.31	724	63579	NC	NC	NC

III. EXTENSION OF THE CONCEPT

In this section, we show how the algorithm presented in the previous section can be adapted to a more general framework. Further extensions are discussed in the extended version of this paper.

A. The stochastic case

Let us assume now that at any time $i \in \{1, \ldots, n\}$, the upper and lower power limits can be decreased because of e.g. a failure from part of the system or a few minutes of required contribution to the reserve. A mathematical translation of this may be to assume that lbP_i and ubP_i are unknown when the decision on x is made, with independent error terms respectively $\sigma_i^- \xi_i^-$ and $\sigma_i^+ \xi_i^+$ and with Φ_- (resp. Φ_+) the cumulative distribution function of ξ_i^- (resp Φ_-). In this case if α is an accepted rate of error, one may want to solve :

$$\begin{aligned} \min \quad & \textstyle\sum_{i=1}^n C_i(x_i) && (7) \\ s.t. \quad & \forall i = 1, \ldots, n, \;\; lbC_i \leq \textstyle\sum_{j=1}^i x_j \leq ubC_i \\ & \alpha \leq P\left(lbP_i + \sigma_i^- \xi_i^- \leq x_i \leq ubP_i + \sigma_+^- \xi_i^+\right) && (8) \end{aligned}$$

In this case D_i, $i = 1, \ldots, n$ can be similarly defined and the recurrence relation given by Equation 3 becomes

$$\begin{aligned} D_i(z) = \quad \min \quad & C_i(x) + D_{i-1}(z - x) \\ s.t. \quad & x \in I_i \\ & lbC_i \leq z - x \leq ubC_i \end{aligned} \tag{9}$$

where for all $i = 1 \ldots, n$ I_i is the set of $u \in \mathbb{R}$ solutions of

$$\left(1 - \Phi_-\left(\frac{u - lbP_{i-1}}{\sigma_{i-1}^-}\right)\right) \Phi_+\left(\frac{u - ubP_{i-1}}{\sigma_{i-1}^+}\right) \geq \alpha$$

and can be computed easily from the knowledge of $\sigma_{i-1}^+, \sigma_{k-1}^-, \alpha, lbC_{i-1}, ubC_{i-1}$. Note that in this case a necessary condition is that for all $i = 1 \ldots, n$, I_i is convex.

B. Extension to the multivariate case

The extension of the optimization problem corresponding to Equation 1 to the more general case when for all $i = 1, \ldots, n$, x_i can be multivariate (i.e. $d-$dimensional vectors, $d \geq 1$) is relatively straightforward. The subsequent optimization problem is as follows:

$$\begin{aligned} \min \quad & \textstyle\sum_{i=1}^n C_i(x_i) && (10) \\ s.t. \quad & \begin{cases} x_i \in \mathcal{P}_i \subset \mathbb{R}^d & i = 1, .., n \\ \sum_{j=1}^i x_j \in \mathcal{Q}_i \subset \mathbb{R}^d & i = 1, .., n \end{cases} \end{aligned}$$

where $\mathcal{P}_i$ and $\mathcal{Q}_i$ are convex Nef Polyhedron of $\mathbb{R}^d$, i.e. convex polyhedron generated by hyperplane separating partitions (see e.g. [11]). The first recurrence relation (the multidimensional counterpart of Equation 3) is :

$$D_k(z) = \min\left(C_k(x) + D_{k-1}(z-x)\right) \quad (11)$$
$$s.t. \begin{cases} x \in \mathcal{P}_k \\ z - x \in \mathcal{Q}_{k-1} \end{cases}$$

Algorithm 1 as presented in subsection II-B still applies if fast algorithm for the Epi-Sum, the sum and a multidimensional version of the squeeze (which is simply the truncation of a convex Nef polyhedron by an hyperplan) exist. However if the procedure to compute either the Epi-sum of the sum or the Legendre transform can be extended with a controlled algorithmic cost for $d \leq 3$, the curse of dimensionality rapidly makes it computationally extensive for $d > 3$. The algorithmic and implementations issues are discussed after one motivating example for the case where $d \leq 3$ to solve a storage problem with market coupling. Another one with temporal coupling is given in the extended version of the paper.

1) Storage and market coupling: In the case where 2 storages A and B, connected to two different markets, are linked together by a line with power limit P_{AB}, the subsequent optimization problem can be solved with the same techniques in dimension 3. If x_i^{AB} is the power flowing from A to B at time i, this indeed leads us to a problem of the following form:

$$\min \sum_{i=1}^{n} C_i(x_i^A, x_i^B, x_i^{AB}) \quad (12)$$
$$s.t \;\; lb_i \leq x_i^s + \epsilon_s x_i^{AB} \leq ub_i \quad i = 1, \ldots, n \;\; s = A, B$$
$$-C_{AB} \leq x_i^{AB} \leq C_{AB} \qquad i = 1, \ldots, n$$
$$lbC_i \leq \sum_{j=1}^{i} x_j^s + \epsilon_s x_j^{AB} \leq ubC_i \quad i = 1, \ldots, n \;\; s = A, B$$

where $\epsilon_A = -1$ $\epsilon_B = 1$ and

$$C_i(x_i^A, x_i^B, x_i^{AB}) = C_i^A(x_i^A - x_i^{AB}) + C_i^B(x_i^B + x_i^{AB})$$

2) The multivariate case general form and computational geometric algorithmic: Fast algorithms to manipulate piecewise linear-quadratic bivariate functions have recently been proposed in [12] for convex hull computation and more recently in [13] for the Legendre transform. They both rely on the decomposition of $\mathbb{R}^2$ induced by hyperplan, i.e. a partition using convex Nef polyhedron. Efficient implementation of 3 dimensional Nef polyhedron has been proposed in [14] and [15] and is now implemented in the computational geometric library CGAL [16]. Integrating the implementation of CGAL into our software to propose algorithms for solving the optimization problem of Equation 10 in the case when $d = 2, 3$ will be the purpose of further work.

IV. Application example : the value of storage for energy displacement in the local price market PJM

In this section we propose to show that the tool presented in the previous section can be used to evaluate rapidly the future yearly income of energy storage in a local price market such as PJM, as a function of its position and dimensions.

A. Storage positioning and dimensioning in the PJM American market

Here, we are interested in studying the operations of a storage operator buying and selling energy on the electricity market (i.e. respectively storing and consuming or discharging and generating electricity). To carry out this optimization, the cost functions i.e. the functions representing the storage operator's gains or losses according to the level of production have to be determined.

We consider that the storage operator is a price-maker, i.e. that the electricity bought (or sold) by the operator increases (respectively decreases) the market price. Therefore, we assume that at a given node the prices are linear functions of the local demand (including the storage effect). This assumption and the subsequent model are described further in the extended version of this paper. The optimization problem for maximizing the benefit of the storage at any given node belongs to the class defined by Equation 1 with quadratic cost function of the form $S * \pi(S)$ where S is the power delivered by storage and $\pi(S)$ is the cost of delivered power (which is a linear function of S). The prices are determined empirically on each node from historical demand data, and prices data with a linear regression. One of the corresponding linear regression is represented in Figure IV-A. It allows, for each node n, to obtain the price sensitivity δ^n (dollard/MW) with respect to the demand modification induced by the use of the storage itself. In the end, for a given time i the quadratic cost function is $C_i^n(x_i) = x_i * (\pi_i^n + \delta^n x_i)$ (where π_i^n is the spot price at node n and time i).

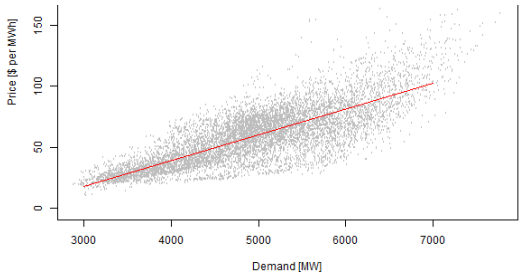

Fig. 1. Scatter plot of the Market prices according to the demand levels

Based on these cost functions that aim at modelling the market revenue of the storage, we calculate the storage unit policy which enables the highest revenue in each of the 93 nodes of the PJM market. Results are presented in Figure IV-A. The number of configuration - localisation and dimension in term of capacity and power limit - is very large (100 localisations, 100*100 dimensions) but our algorithm takes a few milliseconds to compute the revenue for each of these configurations.

This estimation procedure assumes that the prices and their behaviour with respect to the storage participation level are known in advance. It can be shown however that a more complex but realistic procedure gives similar incomes and can also be pursued with our optimization tool. This more complex procedure relies on the evaluation of a storage policy at the scale of month at the beginning of the year, then at the week

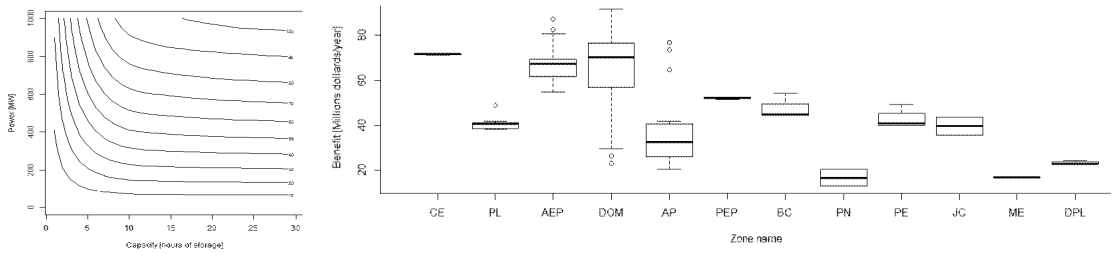

Fig. 2. Benefit opportunities for storage in each PJM zone at fixed capacity and power limit (right) and for one selected zone as a function of capacity and power limit (left). Lines on the left hand side figure range from 10 to 100 Million dollars per year.

scale at the beginning of the month, at the daily scale at the beginning of the week etc. While the price prediction at the hourly time scale are not possible a year in advance, their monthly average are more predictable etc. This kind of more complex multi-scale policy allows to recover similar incomes than that obtain with the simple unrealistic procedure (the difference of revenue being within 5% to 10%).

In any case, the obtained revenues are upper bound of revenues that could be obtained in an operational framework and it is clear that none of the revenues obtained are large enough to justify an investment of storage. In addition the life length of storage would necessitate that the kind of spread in the prices that is observed now is not going to decrease in the next 40 years.

V. Conclusion

In this paper we have proposed a new algorithm to solve efficiently and with an exact solution a class of optimization problems including the one of storage operation with potentially a very long horizon. The proposed implementation is freely available through the R software [6]. In practice, if one wishes to balance prices in the electricity market, this allows to find a solution in less than a second even for an horizon of several years at a hourly resolution. Our implementation handles quadratic cost functions and a certain class of stochastic optimization problems. We have shown how the proposed algorithm extends with the same efficiency to the case of a multivariate state space as long as this space has a low dimension (i.e. not greater than 3) and we have provided application examples to the case of operation of two storage interconnected with possibly congested line.

This efficient algorithm can be used to evaluate rapidly an upper bound on the return on a storage investment in a lot of different configurations.

Acknowledgment

This work was done under the partial funding of French National Research agency (ANR) in the frame of the WinPower project.

References

[1] M. Korpaas, A. T. Holen, and R. Hildrum, "Operation and sizing of energy storage for wind power plants in a market system," *International Journal of Electrical Power & Energy Systems*, vol. 25, no. 8, pp. 599 – 606, 2003, 14th Power Systems Computation Conference. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0142061503000164

[2] D. Maly and K. S. Kwan, "Optimal battery energy storage system (bess) charge scheduling with dynamic programming," *Science, Measurement and Technology, IEE Proceedings* -, vol. 142, no. 6, pp. 453–458, 1995.

[3] R. Ferrero, J. F. Rivera, and S. M. Shahidehpour, "A dynamic programming two-stage algorithm for long-term hydrothermal scheduling of multireservoir systems," *Power Systems, IEEE Transactions on*, vol. 13, no. 4, pp. 1534–1540, 1998.

[4] P. Brown, J. Peas Lopes, and M. Matos, "Optimization of pumped storage capacity in an isolated power system with large renewable penetration," *Power Systems, IEEE Transactions on*, vol. 23, no. 2, pp. 523–531, 2008.

[5] P. Tseng and Z.-Q. Luo, "On computing the nested sums and infimal convolutions of convex piecewise-linear functions," *Journal of Algorithms*, vol. 21, no. 2, pp. 240 – 266, 1996. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0196677496900450

[6] R. D. C. Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2011, ISBN 3-900051-07-0. [Online]. Available: http://www.R-project.org/

[7] R. Bellman, "The theory of dynamic programming." *Bull. Am. Math. Soc.*, vol. 60, pp. 503–515, 1954.

[8] L. Bayn, J. Grau, M. Ruiz, and P. Surez, "A quasi-linear algorithm for calculating the infimal convolution of convex quadratic functions," *Journal of Computational and Applied Mathematics*, vol. 236, no. 12, pp. 2990 – 2997, 2012, ¡ce:title¿Computational methods in Economic modeling and EngineeringCMMSE 2010¡/ce:title¿. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377042711001889

[9] R. T. Rockafellar, *Convex Analysis (Princeton Landmarks in Mathematics and Physics)*. Princeton University Press, Dec. 1996. [Online]. Available: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20\&path=ASIN/0691015864

[10] ——, *Network Flows and Monotropic Optimization,*. John Wiley amd Sons, New York, 1984.

[11] H. Bieri, "Nef polyhedra: A brief introduction," in *Geometric Modelling*, ser. Computing Supplement, H. Hagen, G. Farin, and H. Noltemeier, Eds. Springer Vienna, 1995, vol. 10, pp. 43–60. [Online]. Available: http://dx.doi.org/10.1007/978-3-7091-7584-2_3

[12] B. Gardiner and Y. Lucet, "Convex hull algorithms for piecewise linear-quadratic functions in computational convex analysis," *Set-Valued and Variational Analysis*, vol. 18, no. 3-4, pp. 467–482, 2010. [Online]. Available: http://dx.doi.org/10.1007/s11228-010-0157-5

[13] ——, "Computing the conjugate of convex piecewise linear-quadratic bivariate functions," *Mathematical Programming*, pp. 1–24, 2013. [Online]. Available: http://dx.doi.org/10.1007/s10107-013-0666-8

[14] M. Granados, P. Hachenberger, S. Hert, L. Kettner, K. Mehlhorn, and M. Seel, "Boolean operations on 3d selective nef complexes: Data structure, algorithms, and implementation," in *IN PROC. 11TH ANNU. EURO. SYMPOS. ALG., VOLUME 2832 OF LNCS*. Springer, 2003, pp. 654–666.

[15] P. Hachenberger and et al., "Boolean operations on 3d selective nef complexes: Optimized implementation and experiments," 2005.

[16] "CGAL, Computational Geometry Algorithms Library," http://www.cgal.org.