Discrete Convex Optimization Solvers and Demonstration Softwares*

Nobuyuki Tsuchimura†, Satoko Moriguchi‡, and Kazuo Murota§

Abstract

In the last decade, efficient discrete optimization algorithms have been proposed in discrete convex analysis, which is a unified framework of discrete convex optimization based on the theory of matroids and submodular functions. With a view to disseminating these theoretical results in application fields, we have developed softwares and web applications of fundamental algorithms for discrete convex minimization.

1 Introduction

Nonlinear optimization problems in integer variables appear in many problems in real applications, but they are recognized to be highly difficult to solve and no general-purpose solvers are available for them. In pursuit of better understanding of tractability of nonlinear integer optimization problems, various concepts of discrete convex functions have been proposed together with theoretical investigations. Among others, the framework of discrete convex analysis, which successfully extends the theory of matroids and submodular functions, has attracted attention since the 1990's and many results have already been accumulated in the literature [2, 14–16, 19]. In discrete convex analysis, two kinds of convexities, called L-convexity and M-convexity, are distinguished and are shown to be conjugate to each other. Efficient algorithms are developed for minimizing L-convex functions and M-convex functions.

Discrete convex analysis has also been used in operations research (OR). Specifically, applications to inventory problems and scheduling problems are reported in [1, 7, 16, 21]. Inventory theory, though classical in OR, still plays an important role as the foundation of the modern SCM (Supply Chain Management). In inventory theory discrete convex functions appeared as early as in the 1970's, when Miller [8] introduced a concept of discrete convex functions, called "Miller's convex functions" today, in his study of reparable inventory systems. It has turned out only recently that the function treated by Miller in reparable inventory systems is in fact an $L^\natural$-convex function, which fact implies that the problem formulated by Miller admits a polynomial-time solution algorithm. On the other hand, call centers are

*This is a translation of the paper: N. Tsuchimura, S. Moriguchi, and K. Murota: Discrete convex optimization solvers and demonstration softwares, Transactions of the Japan Society for Industrial and Applied Mathematics, Vol. 23 (2013), No.2, pp. 233–252 (in Japanese).

†Kwansei Gakuin University

‡Advanced Institute of Industrial Technology

§University of Tokyo

getting more and more important as an interface between companies and customers. Koole–Sluis [7] made use of a discrete convexity concept called multimodularity, to deal with a shift scheduling problem of minimizing the number of agents while maintaining an overall service level objective. Multimodularity is known to be equivalent to $\mathrm{L}^\natural$-convexity through a simple transformation of variables.

With a view to disseminating the theoretical results of discrete convex analysis to application fields, we have developed softwares and web applications of fundamental algorithms for discrete convex minimization. We aim at providing an environment that facilitates theoretical and practical works related to discrete convex analysis without full understanding of the technical details of the algorithms. The softwares made available include solvers for discrete convex function minimization, online solvers for interactive minimization of discrete quadratic functions, web applications for an inventory problem and a call center shift scheduling. These are reported in this paper.

2 Discrete Convex Functions

In this section we introduce the concepts of L-convex and M-convex functions, which play the central role in discrete convex analysis, and briefly describe their minimization algorithms. The reader is referred to [14–16] for details.

2.1 L-convex functions and $\mathrm{L}^\natural$-convex functions

For vectors $x, y \in \mathbf{Z}^n$ we denote the vectors of componentwise maximum and minimum by $x \vee y$ and $x \wedge y$, respectively. Let $\mathbf{1} = (1, 1, \ldots, 1) \in \mathbf{Z}^n$. A function $f : \mathbf{Z}^n \to \mathbf{R} \cup \{+\infty\}$ is said to be *L-convex* if it satisfies the following two conditions:

$$f(x) + f(y) \geq f(x \vee y) + f(x \wedge y) \qquad (x, y \in \mathbf{Z}^n), \tag{2.1}$$

$$\exists r \in \mathbf{R} : f(x + \mathbf{1}) = f(x) + r \qquad (x \in \mathbf{Z}^n). \tag{2.2}$$

The inequality (2.1) expresses submodularity on the integer lattice, and, by convention, it is satisfied if $f(x)$ or $f(y)$ is equal to $+\infty$. A function $f : \mathbf{Z}^n \to \mathbf{R} \cup \{+\infty\}$ is said to be *$L^\natural$-convex* if there exists an L-convex function $\tilde{f}(x_0, x_1, \ldots, x_n)$ such that

$$f(x_1, \ldots, x_n) = \tilde{f}(0, x_1, \ldots, x_n). \tag{2.3}$$

The concept of $\mathrm{L}^\natural$-convex functions is equivalent to that of L-convex functions, if the number of variables is not fixed.

Minimizers of $\mathrm{L}^\natural$-convex functions can be characterized by a local condition. For a set $X \subseteq \{1, 2, \ldots, n\}$ we denote its characteristic vector by $\chi_X \in \{0, 1\}^n$, and the effective domain of f is denoted as $\operatorname{dom} f = \{x \in \mathbf{Z}^n \mid f(x) < +\infty\}$.

Theorem 2.1. *For an $L^\natural$-convex function $f : \mathbf{Z}^n \to \mathbf{R} \cup \{+\infty\}$ and a vector $x \in \operatorname{dom} f$, we have*

$$f(x) \leq f(y)\ (\forall y \in \mathbf{Z}^n) \quad \Longleftrightarrow \quad f(x) \leq f(x \pm \chi_X)\ \ (\forall X \subseteq \{1, 2, \ldots, n\}).$$

Testing for the local optimality on the right-hand side above can be reduced to minimizing two submodular set functions[1]

$$\rho_x^+(X) = f(x + \chi_X) - f(x), \qquad \rho_x^-(X) = f(x - \chi_X) - f(x). \tag{2.4}$$

[1] A set function ρ is called *submodular* if it satisfies the inequality $\rho(X) + \rho(X) \geq \rho(X \cap Y) + \rho(X \cup Y)$ for all X and Y [2, 14, 15].

2.2 M-convex functions and M$^\natural$-convex functions

The concept of M-convex functions can be obtained by generalizing the matroid exchange axioms. Let

$$\text{supp}^+(x-y) = \{i \mid x_i > y_i\}, \qquad \text{supp}^-(x-y) = \{j \mid x_j < y_j\} \tag{2.5}$$

and for $i = 1, 2, \ldots, n$, denote the i-th unit vector by χ_i; where $\chi_0 = (0, 0, \ldots, 0)$ by convention.

A function $f : \mathbf{Z}^n \to \mathbf{R} \cup \{+\infty\}$ is called *M-convex* if it satisfies the following exchange axiom:

For any $x, y \in \text{dom}\, f$ and $i \in \text{supp}^+(x-y)$, there exists $j \in \text{supp}^-(x-y)$ such that

$$f(x) + f(y) \geq f(x - \chi_i + \chi_j) + f(y + \chi_i - \chi_j).$$

Since the effective domain of an M-convex function is contained in a hyperplane of a constant component sum, we may project the function along a coordinate axis without essential loss of information about the function values. A function obtained from an M-convex function via a projection is called an M$^\natural$-convex function. That is, a function $f : \mathbf{Z}^n \to \mathbf{R} \cup \{+\infty\}$ is called M$^\natural$-*convex* if the function $\tilde{f}(x_0, x_1, \ldots, x_n)$ defined by

$$\tilde{f}(x_0, x_1, \ldots, x_n) = \begin{cases} f(x) & (x_0 = -\sum_{i=1}^n x_i) \\ +\infty & (x_0 \neq -\sum_{i=1}^n x_i) \end{cases} \tag{2.6}$$

is M-convex. The concept of M$^\natural$-convex functions is equivalent to that of M-convex functions, if the number of variables is not fixed.

Minimizers of M$^\natural$-convex functions can be characterized by a local condition.

Theorem 2.2. *For an* M$^\natural$*-convex function* $f : \mathbf{Z}^n \to \mathbf{R} \cup \{+\infty\}$ *and a vector* $x \in \text{dom}\, f$*, we have*

$$f(x) \leq f(y)\ (\forall y \in \mathbf{Z}^n) \quad \Longleftrightarrow \quad f(x) \leq f(x - \chi_i + \chi_j)\ \ (\forall i, j \in \{0, 1, \ldots, n\}).$$

2.3 Algorithms

Many algorithms have been proposed for minimizing L$^\natural$- or M$^\natural$-convex functions.

2.3.1 Steepest descent methods

Local optimality criteria naturally lead to steepest descent methods for minimization of discrete convex functions. For L$^\natural$-convex functions, Theorem 2.1 yields the following algorithm [6, 13–15].

Steepest descent method for L$^\natural$-convex functions
Step 0: Let x be any vector contained in dom f.
Step 1: Determine $\varepsilon \in \{1, -1\}$ and $X \subseteq \{1, \ldots, n\}$ as follows.
Step 1-1: Let X^+ be (any) minimizer of $\rho_x^+(X) = f(x + \chi_X) - f(x)$.
Step 1-2: Let X^- be (any) minimizer of $\rho_x^-(X) = f(x - \chi_X) - f(x)$.
Step 1-3: Let $(\varepsilon, X) = \begin{cases} (1, X^+) & \text{if } \min \rho_x^+ \leq \min \rho_x^-, \\ (-1, X^-) & \text{if } \min \rho_x^+ > \min \rho_x^-. \end{cases}$

Step 2: If $f(x) \leq f(x + \varepsilon\chi_X)$, then stop ($x$ is a minimizer of f).
Step 3: Let $x := x + \varepsilon\chi_X$ and go to Step 1. □

For the local search in Step 1 we may apply any algorithm for submodular function minimization to $\rho_x^+(X)$ and $\rho_x^-(X)$ (see, e.g., [2, 4, 15]). We use the strongly-polynomial combinatorial algorithm of Iwata–Fleischer–Fujishige [5] (IFF algorithm) and the Fujishige–Wolfe algorithm (FW algorithm) that uses the minimum-norm point (see Section 3).

Also for the minimization of $\mathrm{M}^\natural$-convex functions f we can similarly derive a steepest descent method from Theorem 2.2, where the neighborhood for local search needs to be modified according to Theorem 2.2. Several variants of the steepest descent method for $\mathrm{M}^\natural$-convex functions are proposed, including

(i) The basic form to find (i, j) that minimizes $f(x - \chi_i + \chi_j)$ such as 'descent algorithm' in [14, p. 227] and 'steepest descent algorithm' in [15, pp. 281–283].

(ii) The modified form to find j that minimizes $f(x-\chi_i+\chi_j)$ for an arbitrarily chosen index i, such as 'Modified_Steepest_Descent' in [10].

(iii) A further modification of (ii) that incorporates domain reduction ('Greedy' in [18, page 306]).

2.3.2 Scaling methods

Steepest descent algorithms with the scaling technique, which run in polynomial-time, have been proposed for $\mathrm{L}^\natural$-convex functions and $\mathrm{M}^\natural$-convex functions [10, 14–16, 18]. When given a function $f : \mathbf{Z}^n \to \mathbf{R} \cup \{+\infty\}$, the scaling technique considers the function $f_\alpha : \mathbf{Z}^n \to \mathbf{R} \cup \{+\infty\}$ defined by

$$f_\alpha(x) = f(\alpha x) \qquad (x \in \mathbf{Z}^n), \tag{2.7}$$

where α is a positive integer. The function f_α is naturally regarded as an approximation to f with step size α on the integer lattice, and as such the minimizer of f_α is likely to be located near that of f. On the other hand, f_α is usually easier to minimize than f. Therefore, it is often more efficient to first find a minimizer x_α of f_α and then compute a minimizer of f using x_α as the initial point. Moreover, we can apply the same idea recursively to find the minimizer of f_α. Such an algorithm is called a scaling method. The computational complexity can be analyzed theoretically with the aid of a proximity theorem, which gives an upper bound on the distance between the minimizer x_α of the scaled function f_α and that of the function f.

2.3.3 Continuous relaxation methods

Suppose that, for a discrete convex function $f : \mathbf{Z}^n \to \mathbf{R} \cup \{+\infty\}$, we have a convex function $\bar{f} : \mathbf{R}^n \to \mathbf{R} \cup \{+\infty\}$ in continuous variables such that

$$f(x) = \bar{f}(x) \qquad (\forall x \in \mathbf{Z}^n). \tag{2.8}$$

The continuous relaxation method (e.g., [11, 12]) finds a minimizer $y \in \mathbf{R}^n$ of $\bar{f}$ by using a continuous optimization method, computes an integer vector $x \in \mathbf{Z}^n$ via a rounding of y, and uses x as an initial point for a steepest descent method for f. The computational complexity can be analyzed theoretically with the aid of a proximity theorem, which gives an upper

bound on the distance between the solution y of the continuous relaxation and the minimizer of the function f.

For an $L^\natural$-convex or $M^\natural$-convex function f it is known that a continuous extension $\bar{f}$ in (2.8) exists. In applications, it is often the case that the function f is defined by (2.8) from a function $\bar{f}$ in continuous variables. It is noted that computing $\bar{f}$ from f is easy for $L^\natural$-convex functions, but not for $M^\natural$-convex functions.

3 Our Solver: ODICON

We have developed a solver consisting of discrete convex function minimization algorithms, and named it "ODICON" (Optimization algorithms for DIscrete CONvex functions) [20]. The implemented algorithms are: steepest descent methods (Section 2.3.1), scaling methods (Section 2.3.2), and continuous relaxation methods (Section 2.3.3) for L / $L^\natural$ / M / $M^\natural$-convex functions. ODICON is an open source software in C language, and is expected to be used within other programs rather than used as an independent software. Users who wish to minimize a discrete function should write a C program for that function and call an appropriate program in ODICON.

The programs employ simple implementation of the algorithms and natural interface. In C language there seems to be no unified method to handle dynamic arrays whose size is determined at run time. In view of this, we made particular efforts to minimize the often-encountered difficulty in combining softwares developed by others.

The programs included in our solver are listed in Table 1. Among them is the following:

```
double mgconv_minimize(int dim, double f(int dim, int x[]),
                       int init[], int lower[], int upper[]);
```

which minimizes an $M^\natural$-convex function `f()` in `dim` variables. Users are expected to choose an appropriate program depending on the discrete convexity (L / $L^\natural$ / M / $M^\natural$-convexity) of the target function. The program above starts with the initial vector `init[]`, finds a minimizer (or one of the minimizers in case of multiple minimizers) by the steepest descent method, and returns the minimizer (vector) in `init[]` (while destroying the initial vector) together with the minimum function value as the value of the function subroutine. The search for a minimizer is limited to the range of `lower[`i`]`$\leq$`init[`i`]`$\leq$`upper[`i`]` ($0 \leq i <$ `dim`).

In the above example we need the following declaration statement about the $M^\natural$-convex function to be minimized :

```
double f(int dim, int x[]);
```

which means that the output is of type `double` and the input is a variable of type `int` and an array of type `int`. All programs adopt this format for the function to be minimized.

Suppose, for example, that we want to minimize the following (discrete) $M^\natural$-convex function in three variables:

$$f(x) = x_0^4 + (x_1 - 3)^2 + 5(x_2 - 7)^2. \tag{3.1}$$

This function can be implemented in C as

```
double f(int dim, int x[]) {
    double r = 0;
```

Table 1: Discrete convex function minimization programs implemented in ODICON

L-convex function minimization	(Method of local search)
lconv_minimize	steepest descent method [14](exhaustive enumeration)
lconv_minimize_IFF	steepest descent method [14] (IFF)
lconv_minimize_FW	steepest descent method [14] (FW)
lconv_minimize_scaling	scaling method [14](exhaustive enumeration)
lconv_minimize_scaling_IFF	scaling method [14] (IFF)
lconv_minimize_scaling_FW	scaling method [14] (FW)
lconv_minimize_relax	continuous relaxation method [12] (IFF)
$L^\natural$-convex function minimization	(Method of local search)
lgconv_minimize	steepest descent method [14] (exhaustive enumeration)
lgconv_minimize_IFF	steepest descent method [14] (IFF)
lgconv_minimize_FW	steepest descent method [14] (FW)
lgconv_minimize_scaling	scaling method [14] (exhaustive enumeration)
lgconv_minimize_scaling_IFF	scaling method [14] (IFF)
lgconv_minimize_scaling_FW	scaling method [14] (FW)
lgconv_minimize_relax	continuous relaxation method [12] (IFF)
M-convex function minimization	
mconv_minimize	steepest descent method (i) [14, 15]
mconv_minimize2	steepest descent method (ii) [10]
mconv_minimize3	steepest descent method (iii) [18]
mconv_minimize_scaling	scaling method [10]
mconv_minimize_relax	continuous relaxation method [11]
$M^\natural$-convex function minimization	
mgconv_minimize	steepest descent method (i) [14, 15]
mgconv_minimize2	steepest descent method (ii) [10]
mgconv_minimize3	steepest descent method (iii) [18]
mgconv_minimize_scaling	scaling method [10]
mgconv_minimize_relax	continuous relaxation method [11]

```
    assert(dim == 3);  /* check dim */
    r += x[0] * x[0] * x[0] * x[0];
    r += (x[1] - 3) * (x[1] - 3);
    r += 5 * (x[2] - 7) * (x[2] - 7);
    return r;
}
```

We can minimize this function as follows:

```
int x[3]     = { 0, 0, 0 };
int lower[3] = { -100, -100, -100 };
int upper[3] = {  100,  100,  100 };
double min = mconv_minimize(3, f, x, lower, upper);
```

where the initial point is the origin and the search is made in the range of $-100 \leq x_i \leq 100$. After the execution of the above, the variable `min` contains the minimum function value 0 and the array `x[]` contains a minimizer $(0, 3, 7)$.

For the four kinds of discrete convex functions, L / $L^\natural$ / M / $M^\natural$-convex functions, in discrete variables we have made minimization programs as above. For each type of functions, a variety of minimization algorithms (steepest descent method, scaling method, continuous relaxation method) are implemented, and users can choose whichever they want to use. A common format of the arguments is used to most of the programs, and users can try with different algorithms just by changing the name of the programs.

It is noted, however, that a solver is guaranteed to return a correct answer only if the input function is of the type expected by the algorithm in the solver. That is, users are requested to choose a minimization algorithm that is suitable for the function to be minimized. Otherwise, the solver may return an incorrect answer without trying to detect the possible incorrectness of its output. This is because it is not easy, in general, to check for the convexity type (L / $L^\natural$ / M / $M^\natural$-convexity) of the input function before or during the execution of the algorithm. For a quadratic function, however, we can determine the convexity type, before starting the search for a minimum, by testing for certain properties of the matrix defining the quadratic function (see Section 4.1.1).

For the implementation of minimization algorithms, we need programs for minimization of submodular functions and convex functions in continuous variables. The following programs are used, with some adaptations in the interface to realize a consistent format.

- For submodular function minimization: Iwata–Fleischer–Fujishige (IFF) algorithm [5] implemented by S. Iwata.
- For submodular function minimization: Fujishige–Wolfe (FW) algorithm (minimum-norm base method) implemented by S. Fujishige and S. Isotani [3].
- For continuous minimization: 'L-BFGS', which is an implementation of the quasi-Newton method by J. Nocedal[2], together with a C++ wrapper by T. Kubo[3].
- For random number generation: 'SIMD-oriented Fast Mersenne Twister' implemented by M. Saito and M. Matsumoto[4].

[2] `http://www.ece.northwestern.edu/~nocedal/lbfgs.html`
[3] `http://chasen.org/~taku/software/misc/lbfgs/`
[4] `http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/SFMT/`

The following remarks are in order about the programs listed in Table 1.

- Steepest descent method for L / $L^\natural$-convex functions: Three different programs are provided, which use (i) exhaustive enumeration, (ii) IFF, and (iii) FW as the algorithm of submodular function minimization for local optimization. IFF and FW are subject to rounding errors as they involve floating-point computations (in addition to function evaluations). Exhaustive enumeration, though not a polynomial-time algorithm, is numerically more stable. In submodular function minimization, IFF terminates quickly if the initial solution is close to the minimizer, whereas FW is rather insensitive to the distance of the initial solution to the minimizer.

- Scaling method for L / $L^\natural$-convex functions: Three different programs are provided, which use (i) exhaustive enumeration, (ii) IFF, and (iii) FW as the algorithm of submodular function minimization for local optimization.

- Continuous relaxation method for L / $L^\natural$-convex functions: The IFF-based steepest descent method is chosen to be used at the final stage of the continuous relaxation method. This choice is based on our empirical observation that the integer solution obtained from the rounding of a continuous optimal solution is often sufficiently close to the (true) integral optimal solution, and in such a case, IFF terminates very quickly.

- Steepest descent method for M / $M^\natural$-convex functions: Three different programs are provided, corresponding to the three algorithms mentioned at the end of Section 2.3.1:
  (i) The basic form to find (i, j) that minimizes $f(x - \chi_i + \chi_j)$ [14, 15],
  (ii) The modified form to find j that minimizes $f(x-\chi_i+\chi_j)$ for an arbitrarily chosen[5] index i [10], and
  (iii) A further modification of (ii) that incorporates domain reduction [18].

- Scaling method for M / $M^\natural$-convex functions: The algorithm of Moriguchi–Murota–Shioura [10] is adopted. This algorithm has been proposed as an algorithm with a theoretical guarantee of computational complexity for those M / $M^\natural$-convex functions which retain M / $M^\natural$-convexity after scaling. Nevertheless, the algorithm outputs a correct optimal solution for all M / $M^\natural$-convex functions, though the theoretical guarantee of computational complexity is no longer valid.

- Continuous relaxation method for M / $M^\natural$-convex functions: The program (ii) is chosen in the steepest descent method to be used at the final stage of the continuous relaxation method. The other two choices do not make much difference.

4 Contents of the Website

In our research project [17] we have (i) implemented minimization algorithms for functions with L- and M-convexity and some related algorithms, (ii) made available some optimization solvers for discrete convex functions and other application softwares, and (iii) made demonstration applications for quick experience of the solvers. In this section we describe demonstration applications about discrete convex function minimization solvers, an inventory problem, and a call center shift scheduling problem.

[5] In our implementation we choose i according to the natural ordering of the index.

4.1 Demonstration applications for discrete convex function minimization solvers

For the demonstration of discrete convex function minimization solvers, we have offered web applications that deal with functions such as quadratic $\mathrm{L}^\natural$ / $\mathrm{M}^\natural$-convex functions, quasi-separable $\mathrm{L}^\natural$-convex functions, and laminar $\mathrm{M}^\natural$-convex functions.

4.1.1 Quadratic functions

In the demonstration of $\mathrm{L}^\natural$-convex quadratic function minimization, a user-defined quadratic $\mathrm{L}^\natural$-convex function

$$f(x) = \frac{1}{2}x^\top Ax + b^\top x \tag{4.1}$$

is minimized by the steepest descent method. The user can define the function f through the input of a symmetric matrix $A = (a_{ij})$ of order n and an n-dimensional vector $b = (b_i)$, and specify an initial solution. The application checks for the $\mathrm{L}^\natural$-convexity of the user-defined function[6] (see Fig. 1). If the input is valid (or made valid by the user), "Minimize" button is made available. If this button is pushed, the application applies the steepest descent method (based on exhaustive enumeration) and outputs the result along with the process of computation (Fig. 2). Similarly for $\mathrm{M}^\natural$-convex quadratic function minimization.

4.1.2 Quasi-separable $\mathrm{L}^\natural$-convex functions

In the demonstration of quasi-separable $\mathrm{L}^\natural$-convex function minimization, a function f of the form

$$f(x) = \sum_{i \neq j} f_{ij}(x_i - x_j) + \sum_{i=1}^{n} f_i(x_i) \tag{4.2}$$

is minimized by the steepest descent method. Here f_{ij} and f_i are univariate convex functions, which are quadratic, quartic, exponential, and absolute-value functions in our application. The user can define the function f through the input of parameter values and specify an initial solution. The application applies the steepest descent method (based on IFF) and outputs the result along with the process of computation.

4.1.3 Laminar $\mathrm{M}^\natural$-convex functions

In the demonstration of laminar $\mathrm{M}^\natural$-convex function minimization, a function f of the form

$$f(x) = \sum_{Y \in \mathcal{T}} f_Y(x(Y)), \qquad x(Y) = \sum_{i \in Y} x_i \tag{4.3}$$

is minimized by the steepest descent method. Here $\mathcal{T}$ is a laminar family (a set family such that, for any $X, Y \in \mathcal{T}$, at least one of $X \cap Y$, $X \setminus Y$, $Y \setminus X$ is an empty set), and f_Y's are univariate convex functions, which are quadratic, quartic, exponential, and absolute-value functions in our application. The user can define the function f through the input of parameter values and specify an initial solution. The application applies the steepest descent method (i) and outputs the result along with the process of computation.

[6] f is an $\mathrm{L}^\natural$-convex function $\iff$ $a_{ij} \leq 0$ $(i \neq j)$, $\sum_{j=1}^{n} a_{ij} \geq 0$ $(i = 1, \ldots, n)$.

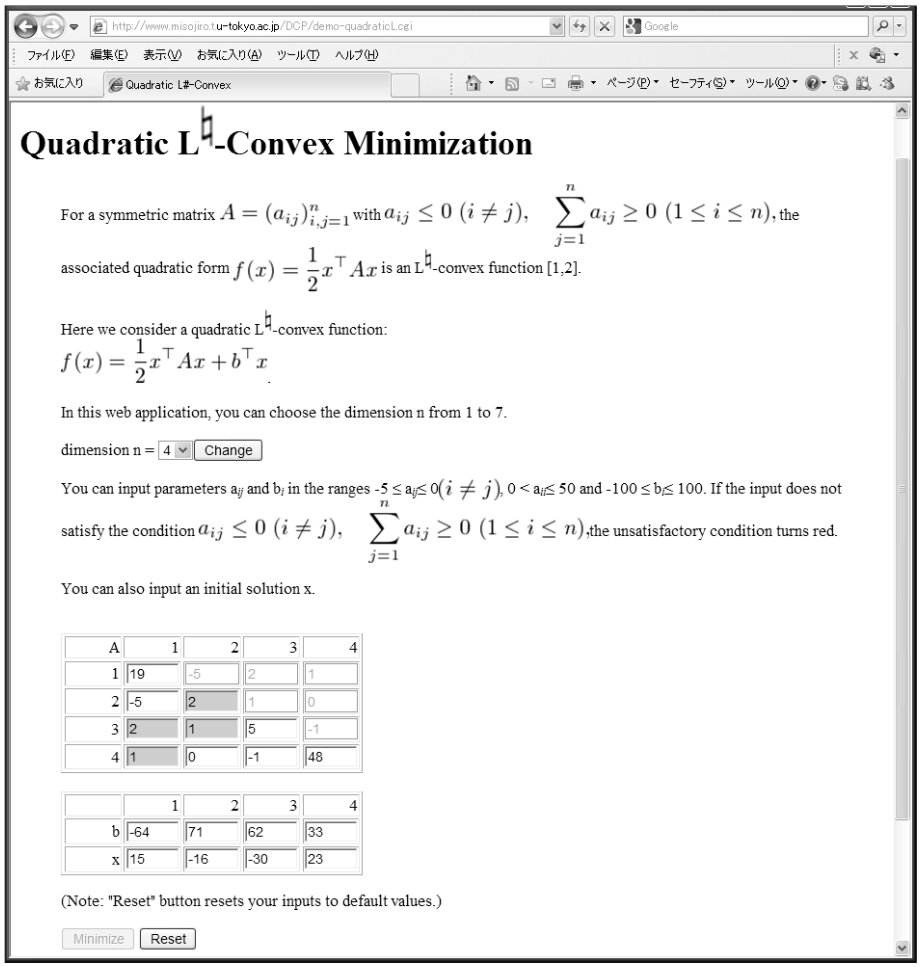

Figure 1: Online solver for quadratic L♮-convex functions (input)

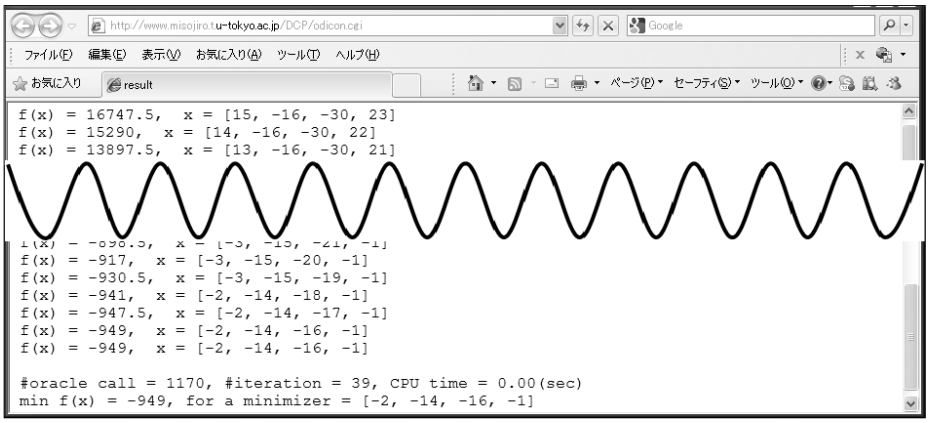

Figure 2: Online solver for quadratic L♮-convex functions (output)

4.2 Application for inventory problem

We made an online application solver for an inventory problem by using an optimization engine based on our discrete convex function minimization solver. In this application the user interactively inputs the number of items n and other parameters p, c_j, λ_j $(j = 1, \ldots, n)$ (see Fig. 3). With the execution of optimization, the optimal numbers of reparable spare parts and the corresponding cost are shown (Fig. 4).

The model for the initial procurement problem in a reparable system, proposed by Miller [8], is a multi-item model with backorders where demand and order quantities take on discrete (integer) values. Such model is used in reparable spares parts maintenance for aircrafts. For mathematical treatment of the problem, the concept of "Miller's convex function" is introduced in [8] together with a minimization algorithm, whose running time is not necessarily bounded by a polynomial in the problem size. It turned out later that the function treated by Miller is in fact an $\mathrm{L}^\natural$-convex function [9, 16].

In Miller's inventory model, the objective is to minimize the total cost consisting of two terms. The first term represents the penalty that is proportional to the steady-state expectation of the maximum number of backorders among n items. The second term is the cost of spares purchases, which is given by $\sum_{j=1}^{n} c_j x_j$ with $c_j > 0$ denoting the unit cost of item j and $x_j \in \mathbf{Z}$ the number of item j to be purchased. We denote by $\varphi_j(m)$ the probability that the demand of the item j (or the nonnegative integer-valued random variable representing this demand) is equal to m. Then the cumulative distribution function of the demand of item j is given by

$$F_j(k) = \sum_{m=0}^{k} \varphi_j(m) \qquad (k \in \mathbf{Z}_+). \tag{4.4}$$

The expected value (in the steady state) of the maximum number of backorders among the n items is given by

$$\sum_{k=0}^{\infty} \left(1 - \prod_{j=1}^{n} F_j(x_j + k) \right) \tag{4.5}$$

(see [16, Section 14.7] for details). The objective is to determine the purchase quantities $(x_1, \ldots, x_n)$ that minimize the total cost

$$f(x) = p \sum_{k=0}^{\infty} \left(1 - \prod_{j=1}^{n} F_j(x_j + k) \right) + \sum_{j=1}^{n} c_j x_j, \tag{4.6}$$

where $p > 0$ represents the penalty for backorders. As the probability distribution of the demand of item j we here assume the Poisson distribution with parameter $\lambda_j > 0$, i.e.,

$$\varphi_j(m) = \exp(-\lambda_j) \frac{\lambda_j^m}{m!} \qquad (m \in \mathbf{Z}_+). \tag{4.7}$$

As already noted, the function f in (4.6) is an $\mathrm{L}^\natural$-convex function, and therefore, we can efficiently minimize $f(x)$ by using $\mathrm{L}^\natural$-convex function minimization algorithms.

This online application software is intended for small-sized sample problems for which the interactive use is appropriate. If the user wishes to try with larger (with n up to 50, say) inventory problems, it is also possible to download our solver ODICON and execute it in the local environment.

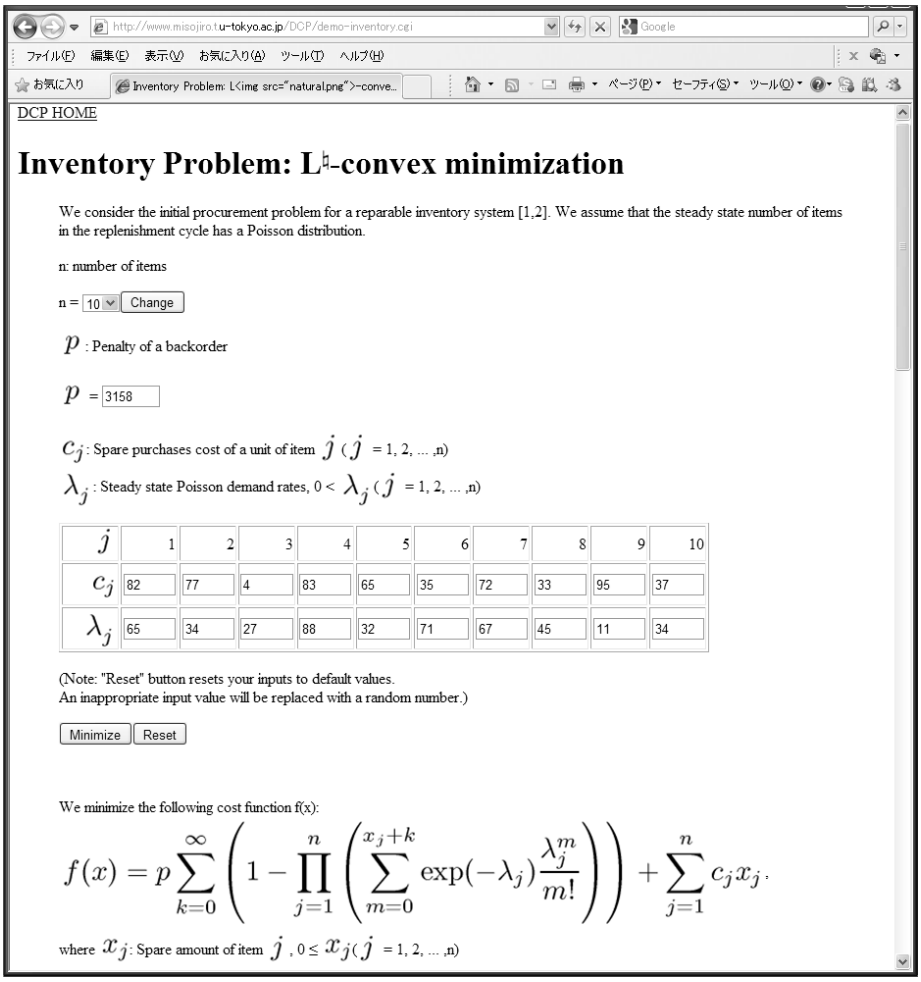

Figure 3: Online solver for an inventory problem (input)

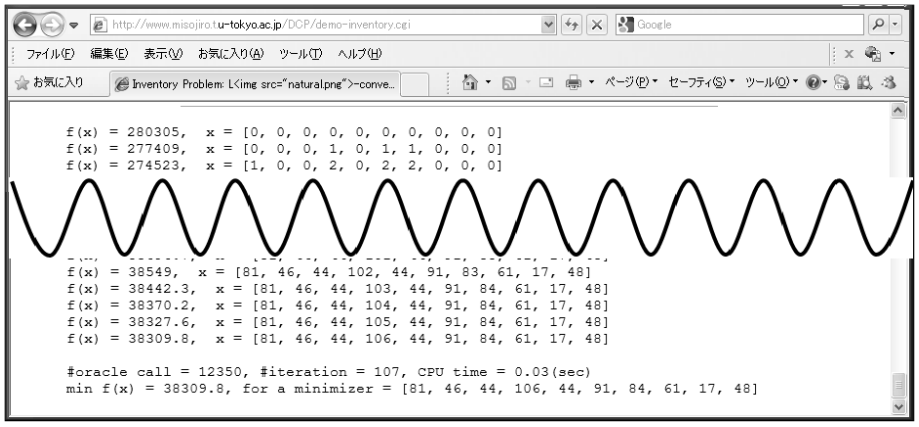

Figure 4: Online solver for an inventory problem (output)

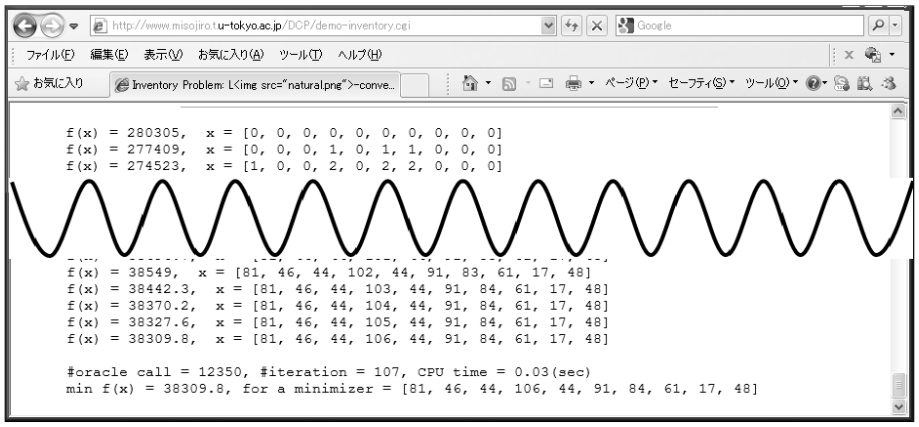

Figure 5: Online solver for a call center shift scheduling problem (input)

4.3 Application for call center shift scheduling

We made an online application solver for a call center shift scheduling problem by using an optimization engine based on our discrete convex function minimization solver. In this application the user interactively inputs customer arrival rate λ_i for each interval i, agent service rate μ, and the threshold c of admissible waiting time (see Fig. 5). With the execution of optimization, the optimal numbers of agents and the corresponding service levels are shown (Fig. 6).

The model of Koole–Sluis [7], which is used here, is as follows.

- The call center is operational during I (consecutive) time intervals, indexed by $i = 1, \ldots, I$.
- Each employee (agent) works for M consecutive intervals.
- During K of the intervals, employees can start working. Shift k starts at the I_k-th interval (and finishes at the beginning of interval $I_k + M$). We assume that $1 \leq I_1 < I_2 < \cdots < I_K \leq I - M + 1$. Figure 7 shows an example of shifts, where $I = 13$, $M = 5$, $K = 4$, $I_1 = 1$, $I_2 = 3$, $I_3 = 6$, $I_4 = 9$.
- For each interval $i = 1, \ldots, I$, there is a function $g_i(n_i)$ representing the service level in terms of the number n_i of agents working in interval i. The function g_i is assumed to be monotone increasing and concave.
- The overall service level S is defined by $S = \sum_{1 \leq i \leq I} g_i(n_i)$.

If y_k employees work in shift k, the number n_i of agents (employees) working in interval i is equal to

$$h_i(y) = \sum_{k: i-M < I_k \leq i} y_k, \tag{4.8}$$

and therefore, the overall service level S is given as

$$S(y) = \sum_{1 \leq i \leq I} g_i(h_i(y)) \qquad (y \in \mathbf{Z}^K) \tag{4.9}$$

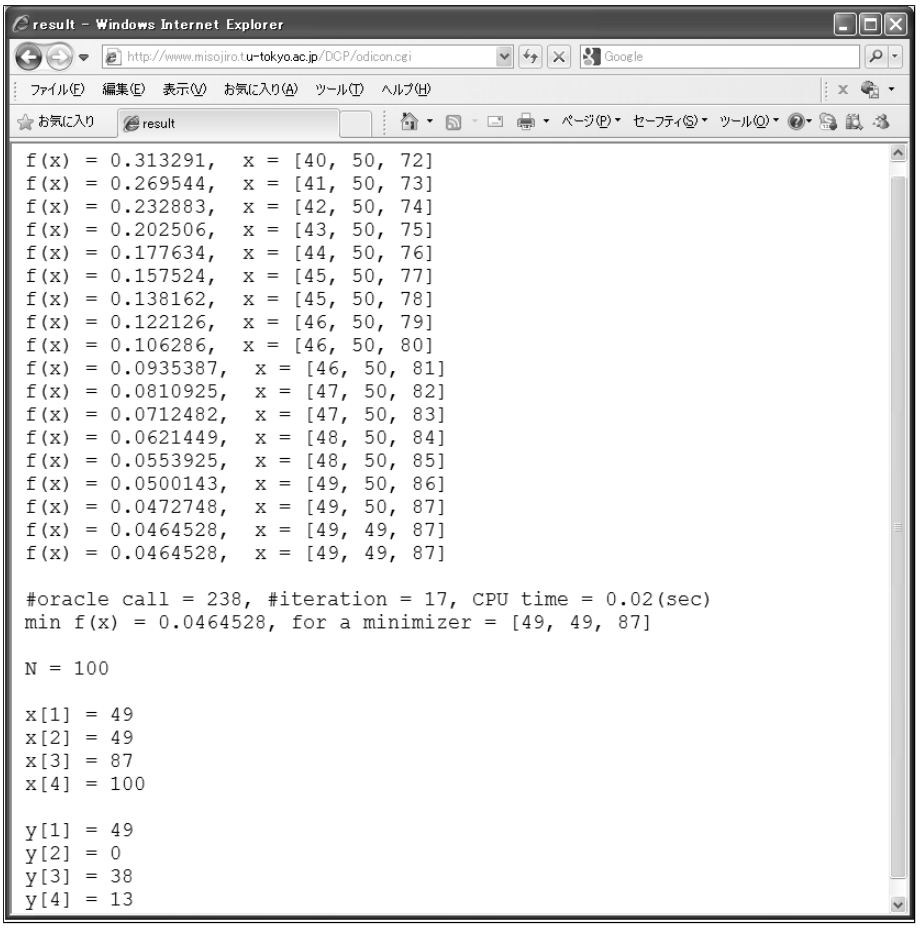

Figure 6: Online solver for a call center shift scheduling problem (output)

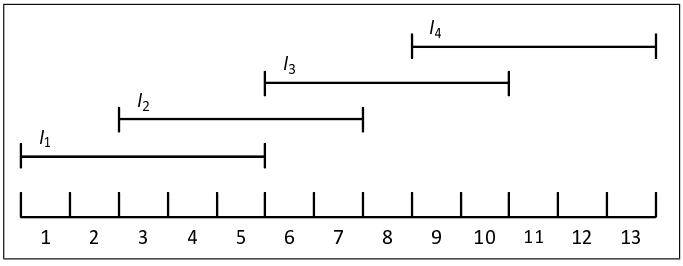

Figure 7: Example of shifts

in terms of $y = (y_1, \ldots, y_K)$. Following [7] we consider the problem of minimizing the number of scheduled employees, under the constraint that the overall service level should at least attain the specified level s:

$$\text{Minimize} \quad \sum_k y_k \quad \text{s.t.} \quad S(y) \geq s, \quad y \in \mathbf{Z}^K. \tag{4.10}$$

To solve this problem we introduce another problem with a parameter N:

$$\text{Maximize} \quad S(y) \quad \text{s.t.} \quad \sum_k y_k = N, \quad y \in \mathbf{Z}^K. \tag{4.11}$$

Then the optimal solution to the problem (4.10) can be obtained by repeatedly solving (4.11) for different N's with the aid of the bisection method to determine an appropriate N.

The concrete form of the overall service level $S(y)$ is determined from the queueing model M/M/n as follows. For $i = 1, \ldots, I$, the customer arrival rate in interval i is denoted by λ_i, whereas the service rate of the agents is denoted by μ. The service level is defined to be the fraction of the customers who do not have to wait longer than c seconds, e.g., $c = 11$ seconds, before getting an agent. Then we have

$$S(y) = \sum_{1 \leq i \leq I} \frac{\lambda_i}{\Lambda} \, \mathrm{P}[W_{\lambda_i}(n_i) \leq c] = \sum_{1 \leq i \leq I} \frac{\lambda_i}{\Lambda} \, \mathrm{P}[W_{\lambda_i}(h_i(y)) \leq c]. \tag{4.12}$$

Here $n_i = h_i(y)$ represents the number of agents in interval i, $\Lambda = \sum_{1 \leq i \leq I} \lambda_i$, $W_\lambda(n)$ denotes the waiting time (random variable) in the queueing model M/M/n (arrival rate λ, service rate μ), and $\mathrm{P}[\cdots]$ means probability. It is known in queueing theory that, in the stationary state, we have

$$\mathrm{P}[W_\lambda(n) \leq c] = 1 - \Pi_n \exp[-n\mu(1 - \rho/n)c], \tag{4.13}$$

where, $\rho = \lambda/\mu$, $\rho/n < 1$, and

$$\Pi_n = \frac{\dfrac{\rho^n}{n!(1-\rho/n)}}{\displaystyle\sum_{k=0}^{n-1} \frac{\rho^k}{k!} + \frac{\rho^n}{n!(1-\rho/n)}}. \tag{4.14}$$

Therefore, for each i, the function[7]

$$g_i(n) = \frac{\lambda_i}{\Lambda} \, \mathrm{P}[W_{\lambda_i}(n) \leq c] \tag{4.15}$$

is increasing and concave in n, satisfying the assumption of our model.

The function $S(y)$ in (4.9) is equipped with discrete concavity. To be specific, under the assumption that each g_i is a monotone increasing concave function for $i = 1, \ldots, I$, the function $-S(y)$ is a kind of discrete convex function, called a multimodular function [7]. Multimodularity is a concept equivalent to $\mathrm{L}^\natural$-convexity; multimodular functions and $\mathrm{L}^\natural$-convex functions are in one-to-one correspondence through a simple transformation of variables [15, 16].

[7] The effective domain of g_i is equal to $\{n \in \mathbf{Z} \mid n > \lambda_i/\mu\}$, and $g_i(n) = -\infty$ for $n \leq \lambda_i/\mu$.

Theorem 4.1. *A function $F : \mathbf{Z}^K \to \mathbf{R} \cup \{+\infty\}$ is multimodular if and only if the function $f : \mathbf{Z}^K \to \mathbf{R} \cup \{+\infty\}$ defined by*

$$f(x) = F(x_1, x_2 - x_1, x_3 - x_2, \ldots, x_K - x_{K-1}) \qquad (x \in \mathbf{Z}^K)$$

is $L^\natural$-convex. In this case we have

$$F(y) = f(y_1, y_1 + y_2, y_1 + y_2 + y_3, \ldots, y_1 + \cdots + y_K) \qquad (y \in \mathbf{Z}^K).$$

This fact reveals that Problem (4.11) is equivalent to an unconstrained minimization of the $L^\natural$-convex function

$$\tilde{f}(x) = -S(x_1, x_2 - x_1, x_3 - x_2, \ldots, x_{N-1} - x_{N-2}, N - x_{K-1}) \qquad (x \in \mathbf{Z}^{K-1}) \tag{4.16}$$

in $K - 1$ variables. Therefore, an exact solution to Problem (4.11) can be found efficiently.

This online application software is intended for small-sized sample problems for which the interactive use is appropriate. If the user wishes to try with other shift structures or larger (with n up to 50, say) scheduling problems, it is also possible to download our solver ODICON and execute it in the local environments.

5 Summary

With a view to disseminating the theoretical results of discrete convex analysis in application fields, we have developed softwares and web applications of fundamental algorithms for discrete convex minimization. We have outlined ODICON, which implements discrete convex function minimization algorithms, and web applications related to inventory and call center shift scheduling.

Acknowledgements

The authors thank Satoru Iwata for providing a program of the IFF method for submodular function minimization and offering useful comments for the development of the solver. They are also grateful to Satoru Fujishige and Shigeo Isotani for providing a program of the FW method for submodular function minimization, and to Naonori Kakimura for corporation in establishing the website. Part of this work is supported by KAKENHI Grant-in-Aid for Scientific Research (B) 21360045 and Grant-in-Aid for Young Scientists (B) 22710148, and also by the Aihara Project, the FIRST program from JSPS, initiated by CSTP.

References

[1] M. Begen and M. Queyranne, Appointment Scheduling with Discrete Random Durations, Mathematics of Operations Research, **36** (2011), 240–257.

[2] S. Fujishige, Submodular Functions and Optimization, 2nd ed., Annals of Discrete Mathematics, **58**, Elsevier, Amsterdam, 2005.

[3] S. Fujishige and S. Isotani, A Submodular Function Minimization Algorithm Based on the Minimum-norm Base, Pacific Journal of Optimization, **7** (2011), 3–17.

[4] S. Iwata, Submodular Function Minimization, Mathematical Programming, **B112** (2007), 45–64.

[5] S. Iwata, L. Fleischer and S. Fujishige, A Combinatorial Strongly Polynomial Algorithm for Minimizing Submodular Functions, Journal of ACM, **48** (2001), 761–777.

[6] V. Kolmogorov and A.Shioura, New Algorithms for Convex Cost Tension Problem with Application to Computer Vision, Discrete Optimization, **6** (2009), 378–393.

[7] G. Koole and E. van der Sluis, Optimal Shift Scheduling with a Global Service Level Constraint, IIE Transactions, **35** (2003), 1049–1055.

[8] B. L. Miller, On Minimizing Nonseparable Functions Defined on the Integers with an Inventory Application, SIAM Journal on Applied Mathematics, **21** (1971), 166–185.

[9] S. Moriguchi and K. Murota, Discrete Hessian Matrix for L-convex Functions, Fundamentals of Electronics, Communications and Computer Sciences, **E88** (2005), 1104–1108.

[10] S. Moriguchi, K. Murota, and A. Shioura, Scaling Algorithms for M-convex Function Minimization, Fundamentals of Electronics, Communications and Computer Sciences, **E85** (2002), 922–929.

[11] S. Moriguchi, A. Shioura, and N. Tsuchimura, M-convex Function Minimization by Continuous Relaxation Approach—Proximity Theorem and Algorithm, SIAM Journal on Optimization, **21** (2011), 633–668.

[12] S. Moriguchi. and N. Tsuchimura, Discrete L-convex Function Minimization Based on Continuous Relaxation, Pacific Journal of Optimization, **5** (2009), 227–236.

[13] K. Murota, Algorithms in Discrete Convex Analysis, IEICE Transactions on Systems and Information, **E83-D** (2000), 344–352.

[14] K, Murota, Discrete Convex Analysis—An Introduction (in Japanese), Kyoritsu Publishing Co., Tokyo, 2001.

[15] K, Murota, Discrete Convex Analysis, SIAM, Philadelphia, 2003.

[16] K, Murota, Primer of Discrete Convex Analysis—Discrete versus Continuous Optimization (in Japanese), Kyoritsu Publishing Co., Tokyo, 2007.

[17] K. Murota, S. Iwata, A. Shioura, S. Moriguchi, N. Tsuchimura, and N. Kakimura, DCP (Discrete Convex Paradigm), `http://www.misojiro.t.u-tokyo.ac.jp/DCP/`

[18] A. Shioura, Fast Scaling Algorithms for M-convex Function Minimization with Application to the Resource Allocation Problem, Discrete Applied Mathematics, **134** (2003), 303–316.

[19] A. Tamura, Discrete Convex Analysis and Game Theory (in Japanese), Asakura Publishing Co., Tokyo, 2009.

[20] N. Tsuchimura, ODICON,
`http://www.misojiro.t.u-tokyo.ac.jp/~tutimura/odicon/`

[21] P. Zipkin, On the Structure of Lost-Sales Inventory Models, Operations Research, **56** (2008), 937–944.