The Data-Correcting Algorithm for the Minimization of Supermodular Functions

Boris Goldengorin • Gerard Sierksma • Gert A. Tijssen • Michael Tso
Department of Econometrics and Operations Research, University of Groningen, P.O. Box 800, 9700 AV Groningen, The Netherlands
Department of Econometrics and Operations Research, University of Groningen, P.O. Box 800, 9700 AV Groningen, The Netherlands
Department of Econometrics and Operations Research, University of Groningen, P.O. Box 800, 9700 AV Groningen, The Netherlands
Department of Mathematics, University of Manchester, Institute of Science and Technology, UMIST, Manchester, United Kingdom
B.Goldengorin@eco.rug.nl

The Data-Correcting (DC) Algorithm is a recursive branch-and-bound type algorithm, in which the data of a given problem instance are "heuristically corrected" at each branching in such a way that the new instance will be as close as possible to polynomially solvable and the result satisfies a prescribed accuracy (the difference between optimal and current solution). In this paper the DC algorithm is applied to determining exact or approximate global minima of supermodular functions. The working of the algorithm is illustrated by an instance of the Simple Plant Location (SPL) Problem. Computational results, obtained for the Quadratic Cost Partition Problem (QCP), show that the DC algorithm outperforms a branch-and-cut algorithm, not only for sparse graphs but also for nonsparse graphs (with density more than 40%), often with speeds 100 times faster.
(*Data-Correcting Algorithm*; *Supermodular Function*; *Global Minimum*)

1. Introduction

Many combinatorial optimization problems have as an underlying model the minimization of a supermodular (or, equivalently, maximization of a submodular) function, among them being the simple plant location (SPL) problem, generalized transportation problems, the max-cut problem with nonnegative edge weights, set covering, and other well known problems involving the minimization of Boolean functions (Nemhauser et al. 1978, Lovasz 1983, Barahona et al. 1988).

Although the general problem of the minimization of a supermodular function is known to be NP-hard, there has been a sustained research effort aimed at developing practical procedures for solving medium and large-scale problems in this class. Often the approach taken has been problem specific, and supermodularity of the underlying objective function has been only implicit to the analysis. For example, Barahona et al. (1988) have addressed the max-cut problem from the point of view of polyhedral combinatorics and developed a branch-and-cut algorithm, suitable for applications in statistical physics and circuit layout design. Beasley (1993) applies Lagrangean heuristics to several classes of location problems including SPL problems and reports results of extensive experiments on a Cray supercomputer. Recently, Lee et al. (1996) have made a study of the quadratic cost partition problem (QCP) of which max-cut with nonnegative edge

0025-1909/99/4511/1539$05.00
1526-5501 electronic ISSN

weights is a special case, again from the standpoint of polyhedral combinatorics.

There have been fewer published attempts to develop algorithms for minimization of a general supermodular function. We believe that the earliest attempt to exploit supermodularity is the work of Petrov and Cherenin (1948), who identified a supermodular structure in their study of railway timetabling. Their procedure was subsequently published by Cherenin (1962) as the "method of successive calculations." Their algorithm however is not widely known in the West (Babayev 1974) where, as far we are aware of, the only general procedures that have been studied in depth are the greedy approximation algorithm from Nemhauser et al. (1978), and the algorithm for maximization of submodular functions subject to linear constraints (Nemhauser and Wolsey 1981). Another greedy approach can be found in Minoux (1977), where an efficient implementation is proposed, known as the "accelerated greedy algorithm" (Robertazzi and Schwartz 1989); it uses a bound already formulated in Khachaturov (1968). For solving the so-called experimental optimal design problem, an accelerated greedy algorithm is applied in Robertazzi and Schwartz (1989), while in Ko et al. (1995) an exact branch and bound type algorithm is developed, which is improved in Lee (1998). In Genkin and Muchnik (1990) an optimal algorithm is constructed with exponential time complexity for the well-known Shannon max-min problem. This algorithm is applied to the maximization of submodular functions subject to a convex set of feasible solutions, and to the problem of what is known as decoding monotonic boolean functions.

In this paper we propose a branch-and-bound procedure for minimizing a general supermodular function that is based on a generalization of an exclusion principle first established in Cherenin (1962). The proposed procedure improves on the greedy algorithm in finding either an exact or an approximate solution to within a prescribed accuracy bound. The approach we take is to develop what we term *Data-Correcting* (*DC*) *algorithms*, which form a class of algorithms, introduced in Goldengorin (1983, 1995) for the solution of NP-hard problems. Crucial in these algorithms is the fact that the data of a given problem instance is "corrected," to obtain a new problem instance belonging to a polynomially solvable class. Actually, the polynomially solvable classes that we use are algorithmically determined.

For example, let us consider the DC algorithm applied to the Traveling Salesman Problem (TSP) with the following algorithmically defined polynomially solvable class. A nonnegative square $n \times n$-matrix $H = \|h(i, j)\|$ is called a *Hungarian matrix* if the Hungarian Algorithm (Nering and Tucker 1993) with input H results in an optimal Hamiltonian cycle; notation $H \in \mathcal{H}$. If a nonnegative square $n \times n$-matrix $C = \|c(i, j)\|$ is not Hungarian, then by correcting some entries of C we will try to find a Hungarian matrix H that is as close as possible to C for some *proximity measure* $\rho(C, H) = \min\{\mu(C, H), \nu(C, H)\}$ of C and H, with $\mu(C, H) = \sum_{i=1}^{n} \max \{|c(i, j) - h(i, j)\| : j = 1, \ldots, n\}$, and $\nu(C, H) = \sum_{j=1}^{n} \max \{|c(i, j) - h(i, j)| : i = 1, \ldots, n\}$. We use the proximity measure $\rho(C, H)$ in the framework of the DC algorithm for finding, by means of a heuristic procedure, an instance $H \in \mathcal{H}$ that is as "close" as possible to C. Usually, this heuristic can be easily constructed by a simple modification of a polynomial algorithm by which we define a polynomially solvable class. In case of the Hungarian algorithm we obtain the so-called Hungarian matrix by patching the subcycles (Lawler et al. 1985).

In the following theorem, which is first published in Goldengorin (1995), it is formulated that the proximity measure is an upper bound for the difference of the lengths of shortest Hamiltonian cycles of C and H, denoted by $\mathrm{OPT}_{\mathrm{TSP}}(C)$ and $\mathrm{OPT}_{\mathrm{TSP}}(H)$, respectively.

Let $C, H \in R^{n \times n}$. Then the following holds: $|\mathrm{OPT}_{\mathrm{TSP}}(C) - \mathrm{OPT}_{\mathrm{TSP}}(H)| \leq \rho(C, H)$. If $\rho(C, H)$ is smaller than a prescribed accuracy ϵ_0, then the TSP solution with respect to H is also a solution with respect to C. Otherwise, the DC algorithm decreases the current value of $\rho(C, H)$ by means of a branching procedure.

Another example can be described as follows. Let Φ be a function defined on a set S and let Ψ belong to a polynomially solvable class of functions which is a subclass of a given class of functions Φ defined also on S, and let $\rho(\Phi, \Psi) = \max \{|\Phi(s) - \Psi(s)|: s \in S\}$ be the

proximity measure. If we denote the optimal values of Φ and Ψ on S by $\Phi^*(S)$ and $\Psi^*(S)$, respectively, then an analogue of the above mentioned inequality is read as follows. Let Φ and Ψ be functions on the set S with optimal values $\Phi^*(S)$ and $\Psi^*(S)$. Then the following holds: $|\Phi^*(S) - \Psi^*(S)| \leq \rho(\Phi, \Psi)$.

Again, if $\rho(\Phi, \Psi)$ is smaller than a prescribed accuracy ϵ_0, then the problem of finding $\Phi^*(S)$ with the given prescribed accuracy ϵ_0 is solved by $\Psi^*(S)$. More information about DC algorithms (general scheme, comparison with branch-and-bound type algorithms, steps of construction, methods, etc.) can be found in Goldengorin (1983, 1995). In this paper we present the DC algorithm for the minimization of supermodular functions.

We have organized our paper as follows. In § 2 we establish two symmetric upper bounds for subproblems of the original problem (Theorem 1) which are the base for constructing two preservation rules (Corollary 1) similar to the ones in Cherenin (1962). We extend the preservation rules to the case where the conditions of Corollary 1 are violated. Corollary 2 is an attempt to explain what we can do in the case when the preservation rules are not applicable. We present the so-called Preliminary Preservation (PP) algorithm, which originally was constructed by Cherenin (1962), and we use it for determining the relevant polynomial solvable class of supermodular functions.

Based on Khachaturov (1968), we are able to present an interesting property of the algorithmically defined polynomially solvable class that is used in the DC algorithm of this paper; this property is in terms of strict local minima of supermodular functions.

In § 3 we describe the main idea of the DC algorithm. Together with Lemma 1 and Theorem 4, we obtain upper bounds for the current accuracy between an optimal and an ϵ-optimal value. The upper bound from Theorem 4 can be incorporated into any branch-and-bound type algorithm. For the supermodular functions case, we introduce a specific correction and show how an upper bound for the difference between ϵ-optimal and optimal values can be calculated. In § 4 we describe the DC algorithm for determining either an exact global minimum or an approximation of a global minimum with prescribed accuracy. In § 5 we illustrate the working of the DC algorithm by means of the SPL problem. We compare our computational results to results from Lee et al. (1996) in § 6. Computer experiments on random instances of the QCP show an improvement upon published results from Lee et al. (1996), particularly when the data correspond to nonsparse graphs. We give some empirical classification on "easy" and "hard" instances of the QCP by using the concept of diagonal dominance. Section 7 gives a number of concluding remarks.

2. Supermodular Functions and the Preliminary Preservation (PP) Algorithm

Let Φ be a real-valued function defined on the power set 2^{I^0} of $I^0 = \{1, 2, \ldots, m\}$; $m \geq 1$. For each $\omega_1, \omega_2 \in 2^{I^0}$ with $\omega_1 \subseteq \omega_2$, define $[\omega_1, \omega_2] = \{\omega \in 2^{I^0} | \omega_1 \subseteq \omega \subseteq \omega_2\}$. Note that $[\varnothing, I^0] = 2^{I^0}$. The interval $[\varphi, I]$ is called a subinterval of $[\varnothing, I^0]$ if $\varnothing \subseteq \varphi \subseteq I \subseteq I^0$; we make use of the notation $[\varphi, I] \subseteq [\varnothing, I^0]$. In this paper we mean by an interval always a subinterval of $[\varnothing, I^0]$. Throughout this paper, it is assumed that Φ attains a finite minimum value on $[\varnothing, I^0]$. The minimum value of the function Φ on the interval $[\varphi, I]$ is denoted by $\Phi^*[\varphi, I]$. For $\epsilon \geq 0$, the problem of *ϵ-minimizing* the function Φ on $[\varphi, I]$ is to find an element $\beta \in [\varphi, I]$ such that $\Phi(\beta) \leq \Phi^*[\varphi, I] + \epsilon$; β is called an *ϵ-minimum* of Φ on $[\varphi, I]$.

The function Φ is called *supermodular* on $[\varphi, I]$ if for each $\beta, \gamma \in [\varphi, I]$ it holds that $\Phi(\beta) + \Phi(\gamma) \leq \Phi(\beta \cup \gamma) + \Phi(\beta \cap \gamma)$. Expressions of the form $I \backslash \{k\}$ and $\varphi \cup \{k\}$ will for conciseness be written as $I - k$ and $\varphi + k$. Let $[\varphi, I]$ be an interval and $k \in I \backslash \varphi$. The following theorem establishes a relationship between the unknown optimal values of Φ on the two parts of the partitioning $[\varphi, I - k]$, $[\varphi + k, I]$ of $[\varphi, I]$. Note that $[\varphi, I - k] \cap [\varphi + k, I] = \varnothing$. The theorem can be used to decide in which part of the partition $[\varphi, I - k]$, $[\varphi + k, I]$ of $[\varphi, I]$ a global minimum of Φ is located.

Theorem 1. *Let Φ be a supermodular function on the interval $[\varphi, I] \subseteq [\varnothing, I^0]$ and let $k \in I \backslash \varphi$. Then the following assertions hold.*

(a). $\Phi^*[\varphi, I - k] - \Phi^*[\varphi + k, I] \leq \Phi(\varphi) - \Phi(\varphi + k)$.

(b). $\Phi^*[\varphi + k, I] - \Phi^*[\varphi, I - k] \leq \Phi(I) - \Phi(I - k)$.

Proof. (a) Let $\gamma \in [\varphi, I - k]$, with $\Phi(\gamma + k) = \Phi^*[\varphi + k, I]$. It then follows from the definition of supermodularity that $\Phi(\varphi + k) + \Phi(\gamma) \leq \Phi(\gamma + k) + \Phi(\varphi)$, which implies that $\Phi(\gamma) \leq \Phi(\gamma + k) + \Phi(\varphi) - \Phi(\varphi + k)$. Hence, $\Phi^*[\varphi, I - k] \leq \Phi(\gamma + k) + \Phi(\varphi) - \Phi(\varphi + k)$. Thus $\Phi^*[\varphi, I - k] - \Phi^*[\varphi + k, I] \leq \Phi(\varphi) - \Phi(\varphi + k)$. The proof of (b) is similar. □

Theorem 1 establishes the conditions for constructing the following rules for detecting subintervals containing at least one global minimum of Φ on $[\varphi, I]$.

Corollary 1. *Let Φ be a supermodular function on the interval $[\varphi, I] \subseteq [\varnothing, I^0]$, and let $k \in I\backslash\varphi$. Then the following assertions hold.*

(a). *First Preservation* (FP) *Rule*:

If $\Phi(\varphi + k) \geq \Phi(\varphi)$, *then* $\Phi^*[\varphi, I] = \Phi^*[\varphi, I - k] \leq \Phi^*[\varphi + k, I]$.

(b). *Second Preservation* (SP) *Rule*:

If $\Phi(I - k) \geq \Phi(I)$, *then* $\Phi^*[\varphi, I] = \Phi^*[\varphi + k, I] \leq \Phi^*[\varphi, I - k]$.

Proof. (a). From Theorem 1 we have that $\Phi^*[\varphi, I - k] - \Phi^*[\varphi + k, I] \leq \Phi(\varphi) - \Phi(\varphi + k)$. By assumption $\Phi(\varphi) - \Phi(\varphi + k) \leq 0$. Hence, $\Phi^*[\varphi, I] = \Phi^*[\varphi, I - k] \leq \Phi^*[\varphi + k, I]$. The proof of 1(b) is similar. □

In Corollary 2 we present an extension of the rules from Corollary 1, appropriate to ϵ-minimization.

Corollary 2. *Let Φ be a supermodular function on the interval $[\varphi, I] \subseteq [\varnothing, I^0]$, and $k \in I\backslash\varphi$. Then the following assertions hold.*

(a). *First θ-Preservation* (θ-FP) *Rule*: *If* $\Phi(\varphi) - \Phi(\varphi + k) = \theta > 0$, *then* $\Phi^*[\varphi, I - k] - \Phi^*[\varphi, I] \leq \theta$, *which means that* $[\varphi, I - k]$ *contains a θ-minimum of* $[\varphi, I]$.

(b). *Second η-Preservation* (η-SP) *Rule*: *If* $\Phi(I) - \Phi(I - k) = \eta > 0$, *then* $\Phi^*[\varphi + k, I] - \Phi^*[\varphi, I] \leq \eta$, *which means that* $[\varphi + k, I]$ *contains a η-minimum of* $[\varphi, I]$.

Proof. The proof of Part (a) is as follows.

Case 1. If $\Phi^*[\varphi, I] = \Phi^*[\varphi, I - k]$, then $\Phi^*[\varphi, I - k] \leq \Phi^*[\varphi, I - k] + \theta$ or $\Phi^*[\varphi, I - k] \leq \Phi^*[\varphi, I] + \theta$.

Case 2. If $\Phi^*[\varphi, I] = \Phi^*[\varphi + k, I]$, then from Theorem 1(a) follows that $\Phi^*[\varphi, I - k] - \Phi^*[\varphi + k, I] \leq \theta$ *or* $\Phi^*[\varphi, I - k] - \Phi^*[\varphi, I] \leq \theta$. The proof of (b) is similar. □

By means of Corollary 1 it is often possible to exclude a large part of $[\varnothing, I^0]$ from consideration when determining a global minimum of Φ on $[\varnothing, I^0]$. The so called *Preliminary Preservation* (PP) *algorithm* determines a subinterval $[\varphi, I]$ of $[\varnothing, I^0]$ that certainly contains a global minimum of Φ, whereas $[\varphi, I]$ cannot be made smaller by using the preservation rules of Corollary 1.

Let $[\varphi, I]$ be an interval. For each $i \in I\backslash\varphi$, define $\delta^+(\varphi, I, i) = \Phi(I) - \Phi(I - i)$ and $\delta^-(\varphi, I, i) = \Phi(\varphi) - \Phi(\varphi + i)$; moreover, define $\delta^+_{\min}(\varphi, I) = \min\{\delta^+(\varphi, I, i) | i \in I\backslash\varphi\}$, $r^+(I, \varphi) = \min\{r | \delta^+(\varphi, I, r) = \delta^+_{\min}(\varphi, I)\}$. Similarly, for $\delta^-(\varphi, I, i)$ define $\delta^-_{\min}(\varphi, I) = \min\{\delta^-(\varphi, I, i) | i \in I\backslash\varphi\}$, $r^-(I, \varphi) = \min\{r | \delta^-(\varphi, I, r) = \delta^-_{\min}(\varphi, I)\}$. If no confusion is likely, we simply write r^-, r^+, δ^-, δ^+ instead of $r^-(I, \varphi)$, $r^+(I, \varphi)$, $\delta^-_{\min}(\varphi, I)$, and $\delta^+_{\min}(\varphi, I)$, respectively.

The Preliminary Preservation Algorithm

Procedure PP(τ, σ; φ, I)

Input: A supermodular function Φ on the subinterval $[\tau, \sigma]$ of $[\varnothing, I^0]$.

Output: A subinterval $[\varphi, I]$ of $[\tau, \sigma]$ such that $\Phi^*[\varphi, I] = \Phi^*[\tau, \sigma]$, $\Phi(\varphi) > \Phi(\varphi + i)$ and $\Phi(I) > \Phi(I - i)$ for each $i \in I\backslash\varphi$.

begin $\varphi := \tau$; $I := \sigma$;
  Step 1: **if** $\varphi = I$
    then goto Step 4;
  Step 2: Calculate δ^+ and r^+;
    if $\delta^+ \leq 0$ {Corollary 1b}
    then begin call PP($\varphi + r^+$, I; φ, I);
      goto Step 4;
    end;
  Step 3: Calculate δ^- and r^-;
    if $\delta^- \leq 0$ {Corollary 1a}
    then begin call PP(φ, $I - r^-$; φ, I);
      goto Step 4;
    end;

Step 4:
end;

Each time either φ or I are updated during the execution of the PP algorithm, the conditions of Corollary 1 remain satisfied, and therefore the invariant $\Phi^*[\varphi, I] = \Phi^*[\tau, \sigma]$ remains valid at each step. At the end of the algorithm we have that $\min\{\delta^+, \delta^-\} > 0$, which shows that $\Phi(\varphi) > \Phi(\varphi + i)$ and $\Phi(I) > \Phi(I - i)$ for each $i \in I \backslash \varphi$. Hence Corollary 1 cannot be applied for further reduction of the interval $[\varphi, I]$ without violating $\Phi^*[\varphi, I] = \Phi^*[\tau, \sigma]$. Note that this remark shows the correctness of the PP algorithm.

In §4 we will use Corollary 1 to make the interval $[\varphi, I] \subseteq [\tau, \sigma]$ even smaller by relaxing the condition $\Phi^*[\varphi, I] = \Phi^*[\tau, \sigma]$; namely, for a prescribed accuracy ϵ we will allow $\Phi^*[\varphi, I] - \Phi^*[\tau, \sigma] \leq \epsilon$.

The following theorem can also be found in Goldengorin (1982). It provides an upper bound for the worst case complexity of the PP algorithm; the complexity function is dependent only on the number of comparisons of values for $\Phi(\omega)$.

Theorem 2. *The time complexity of the PP algorithm is at most* $O(m^2)$.

Note that if the PP algorithm terminates with $\varphi = I$, then φ is a global minimum of Φ on $[\tau, \sigma]$. Any supermodular function Φ on $[\tau, \sigma]$ for which the PP algorithm returns a global minimum of Φ is called a *PP-function*. Note that PP-functions are "algorithmically" determined. In Goldengorin (1982, 1995) an interesting property of PP-functions can be found; it will be explained here shortly.

A subset $L \in [\varnothing, I^0]$ is called a *local minimum* of Φ if for each $i \in I^0$ $\Phi(L) \leq \min\{\Phi(L - i), \Phi(L + i)\}$. We will use the Hasse diagram (Grimaldi 1994) as the ground graph $G = (V, E)$ in which $V = [\varnothing, I^0]$ and a pair (I, J) is an edge iff either $I \subset J$ or $J \subset I$, and $|I \backslash J| + |J \backslash I| = 1$. The graph $G = (V, E)$ is called Φ-*weighted* if the weight of each vertex $I \in V$ is equal to $\Phi(I)$; notation $G = (V, E, \Phi)$.

Let V_0 be the subset of V consisting of all local minima of Φ. Let $H_0 = (V_0, E_0, \Phi)$ be the subgraph of G induced by V_0. This subgraph consists of at least one connected component. We denote the connected components by $H_0^j = (V_0^j, E_0^j, \Phi)$ with $j = 1, 2, \ldots, r$. Note if L_1 and L_2 are vertices in the same component, then $\Phi(L_1) = \Phi(L_2)$. A component H_0^j is called a *component of strict local minima* (in short, STC) if for each $I \in V \backslash V_0^j$, for which there is an edge (I, L) with $L \in V_0^j$, it holds that $\Phi(I) > \Phi(L)$.

A local minimum μ is called a *lower* local minimum if there does not exist another local minimum L with $L \subset \mu$. Similarly, an *upper* local minimum can be defined. It can be easily seen that an STC is a maximal connected set of intervals whose end points are lower and upper local minima. Note that two vertices I and J with $(I, J) \in E$ and $\Phi(I) = \Phi(J)$ are in one STC, if at least one of these vertices is in the STC. In Khachaturov (1968) it is shown that any global minimum is in an STC. This observation implies that we may restrict ourselves to STCs when searching a global minimum of a supermodular function. The following theorem gives a property of PP-functions in terms of STCs; the proof is in Goldengorin (1982).

Theorem 3. *If* Φ *is a supermodular PP-function on* $[\tau, \sigma] \subseteq [\varnothing, I^0]$, *then* $[\tau, \sigma]$ *contains exactly one STC.*

Note that not each supermodular function with exactly one STC on $[\varnothing, I^0]$ is a PP-function. For example, let $m = 3$ and $\Phi(I) = 1$ for any $I \in [\varnothing, \{1, 2, 3\}] \backslash (\{\varnothing\} \cup \{1, 2, 3\})$ and $\Phi(I) = 2$ for any $I \in (\{\varnothing\} \cup \{1, 2, 3\})$. Thus the vertex set of the unique STC defined by this supermodular function can be presented as $[\{1\}, \{1, 2\}] \cup [\{1\}, \{1, 3\}] \cup [\{2\}, \{1, 2\}] \cup [\{2\}, \{2, 3\}] \cup [\{3\}, \{1, 3\}] \cup [\{3\}, \{2, 3\}]$, and the PP-algorithm terminates with $[\varphi, I] = [\varnothing, \{1, 2, 3\}]$. So, Φ is not a PP-function. The supermodular function Φ, of the Simple Plant Location problem in § 5 with the two trivial STCs $\{1, 3\}$, $\{1, 4\}$, is another example of a supermodular function that is not a PP-function.

3. The Main Idea of the DC Algorithm; An Extension of the PP Algorithm

Recall that if a supermodular function Φ is not a PP-function, then the PP algorithm terminates with a

subinterval $[\varphi, I]$ of $[\varnothing, I^0]$ with $\varphi \neq I$ that contains a minimum of Φ without knowing its exact location in $[\varphi, I]$. In this case, the conditions

$$\delta^+(\varphi, I, i) > 0 \quad \text{for } i \in I \backslash \varphi, \tag{1}$$

and

$$\delta^-(\varphi, I, i) > 0 \quad \text{for } i \in I \backslash \varphi \tag{2}$$

are satisfied at termination of the PP algorithm. The basic idea of the DC algorithm is that if a situation occurs for which both (1) and (2) hold, then the data of the current problem will be "corrected" in such a way that a *corrected* function Ψ violates at least one of the Conditions (1) or (2).

In this paper we will restrict ourselves to the situation for which the supermodularity of the corrected function is easy to prove. Hence a situation is studied for which there is an element $i \in I \backslash \varphi$ such that either $\Psi(I - i) \geq \Psi(I)$ or $\Psi(\varphi + i) \geq \Psi(\varphi)$ holds. Now Corollary 1 can be applied again, and we are in the situation that the PP algorithm can be applied. For all possible elements i, we try to choose one for which the correction procedure maintains a solution within the prescribed bound ϵ_0. If such an element i does not exist, we choose an arbitrary $i \in I \backslash \varphi$ and branch the current problem into two subproblems, one on $[\varphi + i, I]$ and one on $[\varphi, I - i]$. We should in any case find answers to the following questions:

- How should the difference between the values of global minima of the corrected and the uncorrected functions be estimated, and how does this difference depend on the specific corrections?
- How should the above mentioned difference be decreased in case it does not satisfy the prescribed accuracy ϵ_0?

The answers to these questions can be found below. To preserve the supermodularity we will use the following correcting rules.

Let $\varnothing \subseteq \varphi \subseteq I \subseteq I^0$, and $r^+, r^- \in I \backslash \varphi$. Moreover, let Φ be a supermodular function on $[\varnothing, I^0]$. For each $\omega \in [\varphi, I]$ define the following two correcting rules.

Correcting Rule 1:

$$\Psi(\omega) = \begin{cases} \Phi(\omega) + \delta^+(\varphi, I, r^+), & \text{if } \omega \in [\varphi, I - r^+]; \\ \Phi(\omega), & \text{otherwise.} \end{cases}$$

Correcting Rule 2:

$$\Psi(\omega) = \begin{cases} \Phi(\omega) + \delta^-(\varphi, I, r^-), & \text{if } \omega \in [\varphi + r^-, I]; \\ \Phi(\omega), & \text{otherwise.} \end{cases}$$

It can be easily seen that if Φ is supermodular on a certain interval, then so is Ψ.

An extension of the PP algorithm is based on the statements of the following lemma.

Lemma 1. *Let Φ be a supermodular function on the interval $[\varphi, I] \subseteq [\varnothing, I^0]$ and let $i \in I \backslash \varphi$. Then*

(a). *If $\Phi(\varphi) - \Phi(\varphi + i) \leq 0$ and $\Phi(\lambda) - \Phi^*[\varphi, I - i] \leq \gamma \leq \epsilon$, then $\Phi(\lambda) - \Phi^*[\varphi, I] \leq \gamma \leq \epsilon$.*

(b). *If $\Phi(I) - \Phi(I - i) \leq 0$ and $\Phi(\lambda) - \Phi^*[\varphi + i, I] \leq \gamma \leq \epsilon$, then $\Phi(\lambda) - \Phi^*[\varphi, I] \leq \gamma \leq \epsilon$.*

(c). *If $0 \leq \delta = \Phi(\varphi) - \Phi(\varphi + i) \leq \epsilon$, and $\Phi(\lambda) - \Phi^*[\varphi, I - i] \leq \gamma \leq \epsilon - \delta$, then $\Phi(\lambda) - \Phi^*[\varphi, I] \leq \gamma + \delta \leq \epsilon$.*

(d). *If $0 \leq \delta = \Phi(I) - \Phi(I - i) \leq \epsilon$, and $\Phi(\lambda) - \Phi^*[\varphi + i, I] \leq \gamma \leq \epsilon - \delta$, then $\Phi(\lambda) - \Phi^*[\varphi, I] \leq \gamma + \delta \leq \epsilon$.*

Proof. The proof of (a) is as follow. From $\delta \leq 0$ and Corollary 1(a) we obtain $\Phi^*[\varphi, I] = \Phi^*[\varphi, I - i]$. Hence $\Phi(\lambda) - \Phi^*[\varphi, I] = \Phi(\lambda) - \Phi^*[\varphi, I - i] \leq \gamma \leq \epsilon$. Since the proof of (b) is similar to that of (a) we conclude with a proof of (c). From $\delta \geq 0$ and Corollary 2(a) we obtain $\Phi^*[\varphi, I - i] - \Phi^*[\varphi, I] \leq \delta$ or $-\Phi^*[\varphi, I] \leq -\Phi^*[\varphi, I - i] + \delta$ or $\Phi(\lambda) - \Phi^*[\varphi, I] \leq \Phi(\lambda) - \Phi^*[\varphi, I - i] + \delta \leq \gamma + \delta \leq \epsilon - \delta + \delta = \epsilon$. □

The following theorem defines the branching step, and shows how a current value of γ can be decreased.

Theorem 4. *Let Φ be an arbitrary function defined on a finite set S and let $S = \bigcup_{t \in P} S_t$ with $\min\{\Phi(\lambda) | \lambda \in S_t\} = \Phi^*(S_t)$, for $t \in P = \{1, \ldots, p\}$. Then for any $\epsilon \geq 0$ the following assertion holds.*

If $\Phi(\lambda_t) - \Phi^*(S_t) \leq \gamma_t \leq \epsilon$ for some $\lambda_t \in S$ and for all $t \in P$, then $\min\{\Phi(\lambda_t) | t \in P\} - \Phi^*(S) \leq \min\{\Phi(\lambda_t) | t \in P\} - \min\{\Phi(\lambda_t) - \gamma_t | t \in P\} = \gamma \leq \max\{\gamma_t | t \in P\} \leq \epsilon$.

Proof. $\min\{\Phi(\lambda_t) | t \in P\} - \Phi^*(S) = \min\{\Phi(\lambda_t) | t \in P\} - \min\{\Phi^*(S_t) | t \in P\} = \min\{\Phi(\lambda_t) | t \in P\} + \max\{-\Phi^*(S_t) | t \in P\} \leq \min\{\Phi(\lambda_t) | t \in P\} + \max\{\gamma_t - \Phi(\lambda_t) | t \in P\} = \min\{\Phi(\lambda_t) | t \in P\} - \min\{\Phi(\lambda_t)$

$- \gamma_t | t \in P\} = \gamma = \min\{\Phi(\lambda_t) | t \in P\} - \min\{\Phi(\lambda_t) | t \in P\} + \max\{\gamma_t | t \in P\} = \max\{\gamma_t | t \in P\} \leq \epsilon$. □

Note that Φ need not be supermodular in Theorem 4. It is clear from the proof of Theorem 4 that γ is independent on the order in which we combine pairs of $\{\Phi(\lambda_t), \gamma_t\}$.

Let us show now that γ may attain $\max\{\gamma_t | t \in P\}$. For sake of simplicity, in case of binary branching, Theorem 4 can be formulated as follows. If $\Phi(\lambda^-) - \Phi^*[\varphi, I - k] \leq \gamma^- \leq \epsilon$, and $\Phi(\lambda^+) - \Phi^*[\varphi + k, I] \leq \gamma^+ \leq \epsilon$, for some $\lambda^-, \lambda^+ \in [\varnothing, I^0]$ and some γ^- and γ^+, then $\min\{\Phi(\lambda^-), \Phi(\lambda^+)\} - \Phi^*[\varphi, I] \leq \min\{\Phi(\lambda^-), \Phi(\lambda^+)\} - \min\{\Phi(\lambda^-) - \gamma^-, \Phi(\lambda^+) - \gamma^+\} = \gamma \leq \max\{\gamma^-, \gamma^+\} \leq \epsilon$.

Now we can construct an example for which $\Phi^*[\varphi, I] = \min\{\Phi(\lambda^+) - \gamma^+, \Phi(\lambda^-) - \gamma^-\}$, and therefore we can assert that the γ in Theorem 4 is best possible. For example, suppose that $\epsilon = 12$, $\Phi(\lambda^-) = 16$, $\gamma^- = 9$, $\Phi^*[\varphi, I - k] = 7$, $\Phi(\lambda^+) = 15$, $\gamma^+ = 8$, and $\Phi^*[\varphi + k, I] = 10$. Then, $\Phi(\lambda^-) - \Phi^*[\varphi, I - k] = 16 - 7 = 9 \leq \gamma^- = 9 < \epsilon$, and $\Phi(\lambda^+) - \Phi^*[\varphi + k, I] = 15 - 10 = 5 < \gamma^+ = 8 < \epsilon$. Moreover, $\Phi^*[\varphi, I] = 7 = \min\{16 - 9, 15 - 8\}$ and $\gamma = \min\{16, 15\} - \min\{16 - 9, 15 - 8\} = 8 < \max\{\gamma^-, \gamma^+\} = \max\{9, 8\} = 9$.

The main step of the DC algorithm, to be formulated in § 4, is called Procedure DC( ). The input parameters of Procedure DC( ) are an interval $[\varphi, I]$, and a prescribed value of ϵ; the output parameters are λ and γ, with $\lambda \in [\varnothing, I^0]$, and $\Phi(\lambda) - \Phi^*[\varphi, I] \leq \gamma \leq \epsilon$. The value of γ is an upper bound for the accuracy of $\Phi(\lambda) - \Phi^*[\varphi, I]$, and may sometimes be smaller than the prescribed accuracy ϵ. The procedure starts with trying to make the interval $[\varphi, I]$ as small as possible by using Corollary 1(a) and 1(b). If this is not possible, the interval is partitioned into two subintervals. Then, with the help of Lemmas 1(c) and 1(d) it may be possible to narrow one of the two subintervals. If this is not possible, the Procedure DC( ) will use the following branching rule.

Branching Rule. For $k \in \text{argmax}\{\max[\delta^-(\varphi, I, i), \delta^+(\varphi, I, i)] | i \in I \backslash \varphi\}$, split the interval $[\varphi, I]$ into two subintervals, $[\varphi + k, I]$ and $[\varphi, I - k]$, and use the prescribed accuracy ϵ of $[\varphi, I]$ for both subintervals.

Our choice for the branching variable $k \in I \backslash \varphi$ is motivated by the observation that $\delta^+(\varphi, I, r^+) \geq \delta^+(\varphi, I - k, r^+)$ and $\delta^-(\varphi, I, r^-) \geq \delta^-(\varphi + k, I, r^-)$, following straightforwardly from the supermodularity of Φ. Hence, the values of δ^+, δ^-, for given r^+, r^-, are seen to decrease monotonically with successive branchings. Our choice is aimed at making the right hand sides δ^+, δ^- as small as possible after branching (and if possible nonpositive), with the purpose of increasing the "probability" of satisfying the preservation rules (see Corollary 1). Moreover, this branching rule makes the upper bound for the difference between a γ-minimum and a global minimum as small as possible.

Note that in Procedure DC( ) λ need not be in the interval $[\varphi, I]$. Notice that in most branch and bound algorithms a solution for a subproblem is searched inside the solution space of that subproblem. From the proofs of Lemma 1 and Theorem 4 it can be seen that this is not necessary here. For any prescribed accuracy ϵ the Procedure DC( ) reads now as follows.

Procedure DC(φ, I, ϵ; λ, γ)

Input: A supermodular function Φ on the interval $[\varphi, I]$, $\epsilon \geq 0$.

Output: $\lambda \in [\varnothing, I^0]$ and $\gamma \geq 0$ such that $\Phi(\lambda) - \Phi^*[\varphi, I] \leq \gamma \leq \epsilon$.

```
begin Step 1: if φ = I
                then begin λ := φ; γ := 0;
                           goto Step 7;
                      end
      Step 2: Calculate δ+ and r+;
              if δ+ ≤ 0
              then begin  call DC(φ + r+, I, ε; λ, γ);
              {Lemma 1b} goto Step 7;
                   end
      Step 3: Calculate δ- and r-;
              if δ- ≤ 0
              then begin call DC(φ, I - r-, ε; λ, γ);
              {Lemma 1a} goto Step 7;
                   end
      Step 4: if δ+ ≤ ε
              then begin call DC(φ + r+, I, ε - δ+; λ, γ);
              γ := γ + δ+ {Lemma 1d};
                           goto Step 7;
                   end
```

Step 5: **if** $\delta^- \leq \epsilon$
then begin call DC(φ, $I - r^-$, $\epsilon - \delta^-$; λ, γ);
$\gamma := \gamma + \delta^-$ {Lemma 1c};
goto Step 7;
end
Step 6: Select $k \in I \backslash \varphi$ (Branching Rule)
call DC($\varphi + k$, I, ϵ; λ^+, γ^+)
call DC(φ, $I - k$, ϵ; λ^-, γ^-)
$\lambda := \text{argmin}\{\Phi(\lambda^-), \Phi(\lambda^+)\}$ {Theorem 4}
$\gamma := \min\{\Phi(\lambda^-), \Phi(\lambda^+)\} - \min\{\Phi(\lambda^+) - \gamma^+, \Phi(\lambda^-) - \gamma^-\}$
Step 7: $\{\Phi(\lambda) - \Phi^*[\varphi, I] \leq \gamma \leq \epsilon\}$
end;

In § 5 we will illustrate this algorithm by solving a Simple Plant Location (SPL) problem.

4. The Data-Correcting (DC) Algorithm

The DC algorithm is a branch-and-bound type algorithm, and is presented as a recursive procedure.

The Data-Correcting Algorithm

Input: A supermodular function Φ on $[\varnothing, I^0]$ and a prescribed accuracy $\epsilon_0 \geq 0$.
Output: $\lambda \in [\varnothing, I^0]$ and $\gamma \geq 0$ such that $\Phi(\lambda) - \Phi^*[\varnothing, I^0] \leq \gamma \leq \epsilon_0$.

begin call DC($\varnothing$, I^0, ϵ_0; λ, γ)
end;

The correctness of the DC algorithm is shown in the following theorem.

Theorem 5. *For any supermodular function Φ defined on the interval $[\varnothing, I^0]$ and for any accuracy $\epsilon_0 \geq 0$, the DC algorithm constructs an element $\lambda \in [\varnothing, I^0]$ and an element $\gamma \geq 0$ such that $\Phi(\lambda) - \Phi^*[\varnothing, I^0] \leq \gamma \leq \epsilon_0$.*

Proof. We only need to show that each step of the DC algorithm is correct. The correctness of Step 1 follows from the fact that if $\varphi = I$ then the interval $[\varphi, I]$ contains a unique solution and λ satisfies the prescribed accuracy ϵ_0 (i.e., $\Phi(\lambda) - \Phi^*[\varphi, I] = \Phi(\lambda) - \Phi(\lambda) = 0 \leq \gamma \leq \epsilon_0$). The correctness of Steps 2 and 3 follow from Lemma 1(b) and Lemma 1(a), respectively; the correctness of Steps 4 and 5 follow from Lemma 1(d) and Lemma 1(c), respectively; the correctness of Step 6 follows from Theorem 4. So, if the Procedure DC( ) is called with the arguments $\varnothing$, I^0, and ϵ_0, then, when it is finished, $\Phi(\lambda) - \Phi^*[\varnothing, I^0] \leq \gamma \leq \epsilon_0$ holds. □

It is possible to make the DC algorithm more efficient if we fathom subproblems by using lower bounds. For subproblems of the form $\min\{\Phi(\omega) | \omega \in [\varphi, I]\} = \Phi^*[\varphi, I]$, the following lemma, due to Khachaturov (1968) and Minoux (1977), provides two lower bounds.

Lemma 2. *If $\Phi(\varphi) - \Phi(\varphi + i) > 0$ and $\Phi(I) - \Phi(I - i) > 0$ for all $i \in I \backslash \varphi$, then $lb_1 = \Phi(\varphi) - \sum_{i \in I \backslash \varphi} [\Phi(\varphi) - \Phi(\varphi + i)] \leq \Phi^*[\varphi, I]$, and $lb_2 = \Phi(I) - \sum_{i \in I \backslash \varphi} [\Phi(I) - \Phi(I - i)] \leq \Phi^*[\varphi, I]$.*

We next explain how to incorporate such a lower bound into the DC algorithm. During the running of the DC program we keep a global variable β in the subset of I^0 that has the lowest function value found so far. Then we can include a Step 3a after Step 3 in Procedure DC( ).

Step 3a: Calculate $lb := \max\{lb_1, lb_2\}$;
if $\Phi(\beta) - lb \leq \epsilon$
then begin $\lambda := \beta$; $\gamma := \Phi(\beta) - lb$;
goto Step 7;
end

It is obvious that λ and γ satisfy $\Phi(\lambda) - \Phi^*[\varphi, I] \leq \Phi(\beta) - lb = \gamma \leq \epsilon$. Note that in this case, β is, in general, not an element of the interval $[\varphi, I]$.

The DC algorithm can also be used as a fast greedy heuristic. If the prescribed accuracy ϵ_0 is very large, branchings never occur at Step 6; the interval $[\varphi, I]$ is halved in every recursive call of the algorithm until $\varphi = I$, and a "greedy" solution is found. Moreover, the calculated accuracy γ gives insight into the quality of the solution obtained, as it is an upper bound for the difference in value of the solution obtained and an optimal solution. Note that, thanks to Steps 2 and 3, the "greedy" solution found by the DC algorithm when a large ϵ is specified, is in general better than one obtained by a standard or accelerated greedy algorithm like the ones described in Minoux (1977).

5. The Simple Plant Location Problem: An Illustration

The DC algorithm is used for the determination of a global minimum (0-minimum) and a 2-minimum for the SPL problem of which the data are presented in Table 1. This example is borrowed from Boffey (1982).

For solving the SPL problem it suffices to solve the problem $\min\{\Phi(\omega)|\omega \in [\varnothing, I^0]\} = \Phi^*[\varnothing, I^0] = \Phi(\alpha)$ with $I^0 = \{1, 2, 3, 4\}$, $n = 5$ and $\Phi(\omega) = \sum_{i\in\omega} r_i + \sum_{j=1}^{n} \min_{i\in\omega} c_{ij}$.

As usual for the SPL problem, r_i is the fixed cost of opening a plant at location i, and c_{ij} is the cost of satisfying the demand of customer j by plant i. The recursive solution trees for the cases $\epsilon_0 = 0$ and $\epsilon_0 = 2$ are depicted in Figure 1 and Figure 2, respectively. Each subproblem is represented by a box in which the values of the input and the output parameters are shown. At each arc of the trees the corresponding steps of the Procedure DC( ) are indicated. In Figure 1 the prescribed accuracy $\epsilon_0 = 0$ is not yet satisfied at the second level of the tree, so that a branching is needed. In the case of $\epsilon_0 = 2$, the DC algorithm applies the branching rule at the third level because after the second level the value of the current accuracy is equal to 1 ($\epsilon = 1$).

6. The Quadratic Cost Partition Problem: Computational Experiments

For given real numbers p_i and nonnegative real numbers q_{ij} with $i, j \in I^0$, the Quadratic Cost Partition Problem (QCP) is the problem of finding a subset T of I^0 such that the weight $\sum_{i\in T} p_i - \frac{1}{2}\sum_{i,j\in T} q_{ij}$ is as large as possible. Let I^0 be the vertex set, $E \subseteq I^0 \times I^0$ the edge set of the edge-weighted graph $G = (I^0, E)$ with $w_{ij} \geq 0$ the edge weights. For each $T \subseteq I^0$, the cut $\delta(T)$ is defined as the edge set for which each edge has one end in T and the other one in $I^0\backslash T$. It is easy to see that the max-cut problem with nonnegative edge weights is a QCP where $p_i = \sum_{j\in I^0} w_{ij}$ and $q_{ij} = 2w_{ij}$, for $i, j \in I^0$.

We have obtained computational results using randomly generated connected graphs with the number of vertices varying from 40 to 80 and edge density d varying from 10% to 100% (i.e., $d \in [0.1, 1.0]$). Here

Table 1 The Data of the SPL Problem

Location		Delivery Cost to Site				
i	r_i	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$
1	7	7	15	10	7	10
2	3	10	17	4	11	22
3	3	16	7	6	18	14
4	6	11	7	6	12	8

Figure 1 The Recursive Solution Tree for $\epsilon_0 = 0$

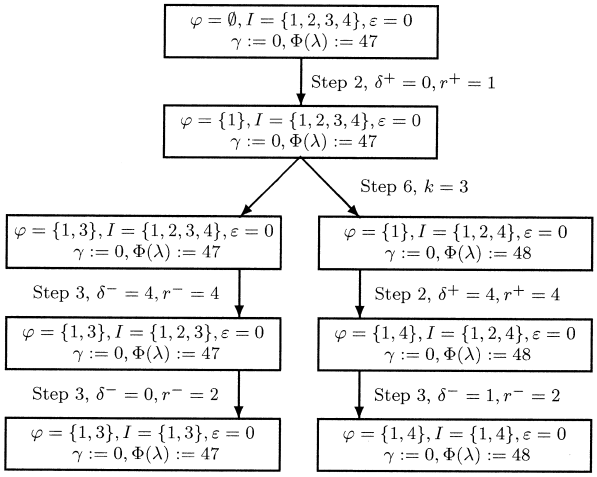
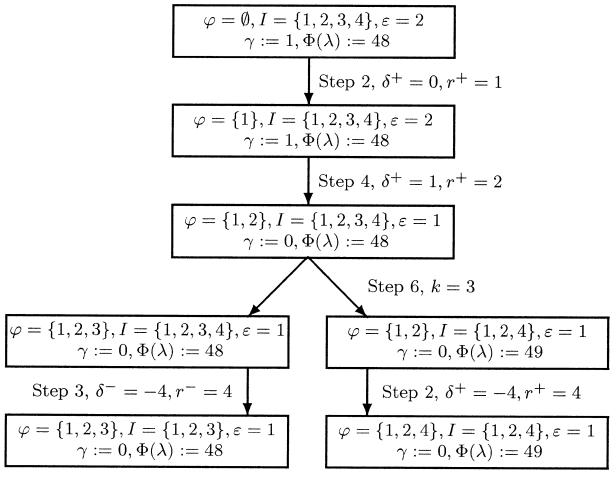


Figure 2 The Recursive Solution Tree for $\epsilon_0 = 2$



the edge density is $d = |E|/\binom{m}{2}$, $|E|$ is the number of random generated edges and $\binom{m}{2}$ is the number of edges in the corresponding simple complete graph. The values of p_i and q_{ij} are uniformly distributed in the intervals [0, 100] and [1, 100], respectively. The computational results are summarized in Table 2. We have tested the DC algorithm on the QCP test problems from Lee et al. (1996), and have made a comparison between our results and those from Lee et al. (1996).

Each problem set is labeled by the number of vertices of the graph together with their densities d. For example, problem 50/7 refers to graphs with 50 vertices and density $d = 0.7$ (or 70%), problem 40 refers to complete graphs with 40 vertices. For each combination of density and number of vertices, five random problems were solved. The column labeled "Lee et al." in Table 2 contains the average computational times for the problems on a RISC 6000 workstation as given in Lee et al. (1996). The DC algorithm was coded by means of Turbo Pascal 6.0 and was executed on a PC with a 133 Mhz processor. Cells with "min.," "avg.," and "max." in Table 2 show minimum, maximum, and average performances of two statistics for the DC algorithm: "the number of generated subproblems" solved, and "the number of fathomed subproblems" indicating the number of subproblems discarded by means of the lower bounds lb_1 and lb_2 from Lemma 2. When the graph has at most 40 vertices, the problem is very easy, and the calculation times are less than 0.05 seconds. For problems with at least 40 vertices the average calculation times grow exponentially with decreasing values of the density d (see Figure 3) for all values of ϵ_0. This behavior differs from the results of the algorithm from Lee et al. (1996); their calculation times grow with increasing densities.

Table 2 The Comparison of Computational Results

	Time Average, Sec		# of Generated Subpr.			# of Fathomed Subpr.		
Prob.	Lee et al.	DC	min.	avg.	max.	min.	avg.	max.
40/2	0.97	0.10	618	797	972	306	396	481
40/3	2.09	0.08	470	640	793	235	313	385
40/4	6.79	0.05	430	539	735	204	258	354
40/5	6.63	0.028	428	497	584	201	231	278
40/6	8.62	0.038	340	387	434	153	173	192
40/7	11.40	0.030	204	216	267	85	100	116
40/8	14.57	0.028	217	261	292	80	95	103
40/9	8.46	0.012	107	154	223	34	42	56
40	13.89	0.004	119	160	213	33	38	48
50/1	0.56	0.29	1354	1885	2525	686	945	1258
50/2	5.36	0.45	2100	2778	3919	1042	1393	1971
50/3	16.19	0.27	1671	2074	2565	814	1019	1268
50/4	95.32	0.18	1183	1576	1976	576	755	950
50/5	38.65	0.08	870	943	1051	414	447	502
50/6	43.01	0.07	646	725	798	291	321	345
50/7	48.07	0.05	610	648	714	245	270	294
60/2	12.11	1.56	5470	8635	11527	2718	4303	5744
60/3	183.02	0.71	3481	5069	7005	1736	2519	3478
60/4	150.50	0.39	2450	3037	3895	1221	1503	1917
60/5	137.22	0.22	1701	2080	2532	825	1012	1236
70/2	437.74	4.89	15823	23953	34998	7909	11971	17486
70/3	367.50	1.91	9559	11105	13968	4769	5540	6967
80/1	20.87	28.12	55517	92836	132447	27771	46418	66228
80/2	864.27	17.10	64261	66460	68372	32102	33202	34160

For problems with density of more than 10%, our algorithm is faster than the algorithm from Lee et al. For one problem (80/1) with density equal to 10% our algorithm uses more time.

Some typical properties of the behavior of the DC algorithm are shown in Figures 3, 4, and 5. In Figure 4 it can be seen that the calculation time of the DC algorithm grows exponentially when the number of vertices increases. This is to be expected since general QCPs are NP-hard. Figure 5 shows how the calculation times of the DC algorithm depend on the value of ϵ_0. We have used different prescribed accuracies varying from 0% to 5% of the value of the global minimum.

In all experiments with $\epsilon_0 > 0$ the maximum value of the calculated γ (denoted by γ_{max}) is at most 0.01949 $\times \Phi^*[\varnothing, I^0]$, i.e., within 2% of a global minimum. Moreover, for all test problems with density at least 30% ($d \geq 0.3$), we obtained $\gamma_{max} = 0$, that is, we found an exact global minimum with a calculation time of at most 5 seconds. In Figure 6, γ_{max} is depicted for various values of ϵ_0.

Figure 3 **Average Calculation Time in Secs. against the Density d (case: m = 80, ϵ_0 = 0)**

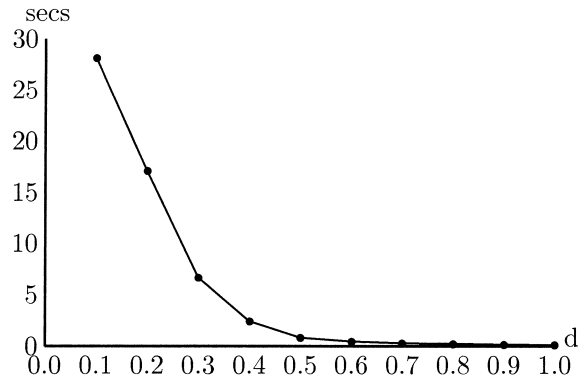


Figure 4 **Average Calculation Time in Secs. against the Number of Vertices m (case: d = 0.3; ϵ_0 = 0)**

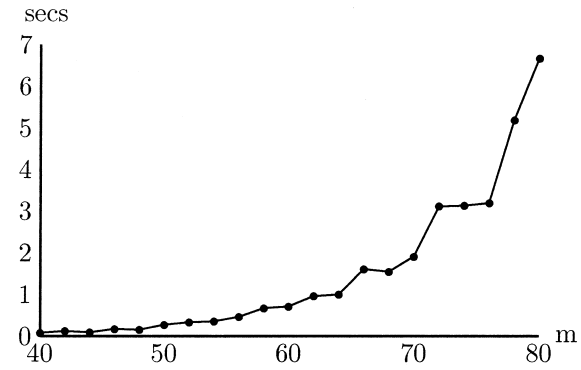


Figure 5 **Average Time in Secs. against Precscribed Accuracy ϵ_0 (case: m = 80, d = 0.2)**

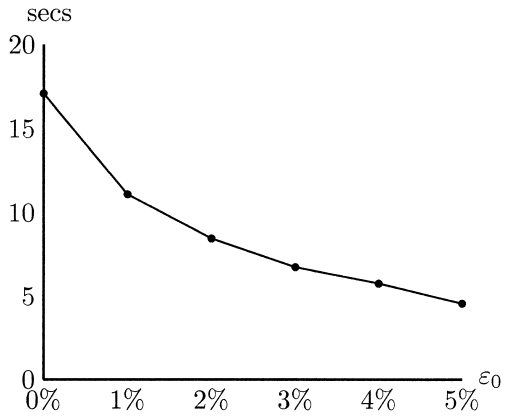
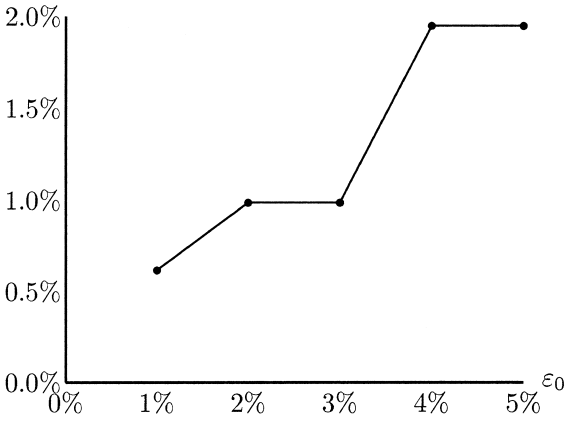


Figure 6 **γ_{max} as Percentage of the Value of a Global Minimum**



One of the referees of this paper suggested that in case of the QCP the diagonal dominance of the matrix might have a great influence on the calculation times. Assuming that all p_i are positive, the *diagonal dominance* (dd) is defined as $dd_i = p_i/(2\sum_{j\neq i} q_{ij})$, $i \in I^0$, i.e., it is the quotient of the "main diagonal entry" p_i and the sum of the off diagonal entries in the ith row and column of $Q = \|q_{ij}\|$. We have calculated the diagonal dominance values of the instances from Table 2. The results of these calculations are presented in Table 3. The first column shows that 78.8% of the

diagonal elements have values in the interval [0.05, 0.2]; the meaning of Columns 2 and 3 are similar.

We have studied the influence of the diagonal dominance on the average calculation time of the DC algorithm for the following randomly generated instances. The number of vertices m varies from 40 to 80, the edge densities d are chosen in the interval [0.1, 1.0], and the edge weights are randomly generated from the interval [1, 100], just as in Table 2. The weights of the vertices p_i however, are calculated from the edge weights by using a constant dd for all vertices in the same instance, namely $p_i = 2dd \sum_{j \neq i} q_{ij}$, $i \in I^0$. The results for the case $m = 40$ are shown in Figure 7.

The calculation times grow exponentially with increasing values of the density d, and for fixed d they grow exponentially if dd comes close to 0.5. The maximum calculation time is attained for $dd = 0.5$ and $d = 1.0$. This is the case of a "pure" max-cut problem on a complete graph. Recall that for the instances from Lee et al. (1996), as shown in Figure 3, the calculation times decrease by increasing values of the density. Notice that this phenomenon does not occur in case of constant dd.

We may conclude that diagonal dominance is a good "yardstick" for measuring the intractability of instances of the QCP. For example, our randomly generated instances with a constant dd for all vertices from the interval union [0.05, 0.2] ∪ [0.8, 1.0] can be classified as "easy" instances of the QCP and from the interval [0.4, 0.6] as "hard" ones. In all our experiments with constant dd, the effect of exponentially increasing calculation times with increasing values of m (see Figure 4), and the exponentially decreasing calculation times with increasing values of ϵ_0 (see Figure 5) is preserved.

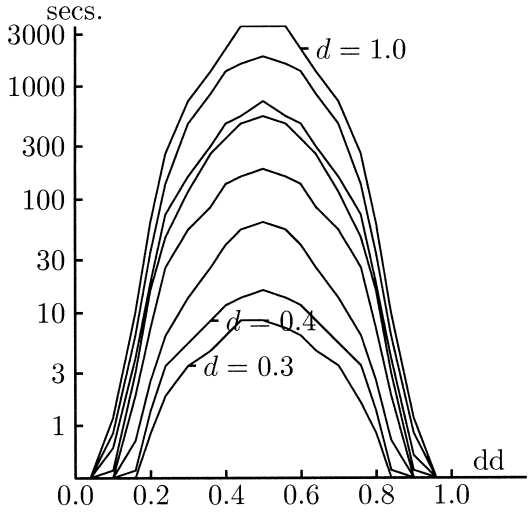
7. Conclusions

Theorem 1 can be considered as the basis of our Data Correcting algorithm. It states that if an interval $[\varphi, I]$ is split into $[\varphi, I - k]$ and $[\varphi + k, I]$, then the difference between the supermodular function values $\Phi(\varphi)$ and $\Phi(\varphi + k)$, or between the values of $\Phi(I)$ and $\Phi(I - k)$ is an upper bound for the difference of the (unknown!) optimal values on the two subintervals. This difference is used for "correcting" the current data (values of Φ) in the algorithm.

Table 3 The Distribution of the Diagonal Dominances

dd	[0.05, 0.2]	[0.2, 0.8]	[0.8, 25]
percentage	78.8	17.2	4.0

Figure 7 Average Time in Secs. against Diagonal Dominance (Cases: m = 40, d = 0.3, 0.4, . . . , 1.0)



Another keystone in the paper is Theorem 4. For any list of subsets S_t, $t \in P$ that cover the feasible region, for example, $S = \bigcup_{t \in P} S_t$, it enables us to derive a new upper bound between an upper bound on the optimal value of the objective function on S and the optimal value. This new and sharper upper bound, when implemented in the DC algorithm, yields an increase of the calculated accuracy, and a decrease in the value of the associated parameter ϵ. Moreover, this bound can also be built into other branch-and-bound type algorithms for increasing the calculated accuracy.

The DC algorithm presented in this paper is a recursive branch-and-bound type algorithm. This recursion makes it possible to present a short proof of its correctness; see Theorem 5.

We have tested the DC algorithm on cases of the QCP, enabling comparison with the results presented in Lee et al. (1996). The striking computational result is the ability of the DC algorithm to find exact solutions

for instances with densities larger than 30% and with a prescribed accuracy of 5% within fractions of second. For example, an exact global optimum of the QCP with 80 vertices and 100% density, was found within 0.22 seconds on a PC with a 133 Mhz processor. We point out that when the value of ϵ_0 is very large, the DC algorithm behaves as a greedy algorithm.

One of the referees has proposed a "measure" of intractability of the QCP in connection with the DC algorithm, namely the so called diagonal dominance. Our computational experiments with the DC algorithm show that instances of the QCP with diagonal dominances from the intervals union $[0.05, 0.2] \cup [0.8, 25]$ can be classified as "easy" instances, and instances with diagonal dominances from the interval [0.4, 0.6] as "hard" to solve.

Finally, we would like to remark that the DC algorithm can be used for broad classes of combinatorial optimization problems that are reducible to the minimization of a supermodular function.[1]

[1] The authors are grateful to the referees for their useful comments which improved the presentation of the results, especially for the introduction of the concept of diagonal dominance.

References

Babayev, Dj. A. 1974. Comments on the note of Frieze. *Math. Programming* **7** 249–252.

Barahona, F., M. Grötschel, M. Jünger, G. Reinelt. 1988. An application of combinatorial optimization to statistical physics and circuit layout design. *Oper. Res.* **36**(3) 493–512.

Beasley, J. E. 1993. Lagrangean heuristics for location problems. *European J. Oper. Res.* **65** 383–399.

Boffey, T. B. 1982. *Graph Theory in Operations Research.* Academic Press, London, U.K.

Cherenin, V. P. 1962. Solving some combinatorial problems of optimal planning by the method of successive calculations. *Proc. Conf. Experiences and Perspectives of the Appl. Math. Methods and Electronic Comput. Planning.* Mimeograph, (in Russian) Novosibirsk, Russia.

Genkin, A. V., I. B. Muchnik. 1990. Optimum algorithm for maximization of submodular functions. *Automation and Remote Control USSR* **51** 1121–1128.

Goldengorin, B. 1982. On the stability of solutions in problems with a single connected component of local minima. *Models and Methods of Solving Problems of Economic Systems Interaction*. (in Russian) Nauka, Novosibirsk 149–160.

——. 1983. A correcting algorithm for solving some discrete optimization problems. *Soviet Math. Doklady* **27** 620–623.

——. 1995. *Requirements of Standards*: *Optimization Models and Algorithms*. Russian Operations Research, Hoogezand, The Netherlands.

Grimaldi, R. P. 1994. *Discrete and Combinatorial Mathematics*: *An Applied Introduction*, 3rd ed. Addison-Wesley Publishing Company, New York.

Khachaturov, V. R. 1968. Some problems of the successive calculation method and its applications to location problems. (In Russian) Ph.D Thesis, Central Economics & Mathematics Institute, Russian Academy of Sciences, Moscow, Russia.

Ko, C.-W., J. Lee, M. Queyranne. 1995. An exact algorithm for maximum entropy sampling. *Oper. Res.* **43** 684–691.

Lawler, E. L., J. K. Lenstra, A. H. G. Rinnooy Kan, D. Shmoys, eds. 1985. *The Traveling Salesman Problem*: *A Guided Tour of Combinatorial Optimization*. Wiley, Chichester, U.K.

Lee, J. 1998. Constrained maximum-entropy sampling. *Oper. Res.* **46** 655–664.

Lee, H., G. L. Nemhauser, Y. Wang. 1996. Maximizing a submodular function by integer programming: Polyhedral results for the quadratic case. *European J. Oper. Res.* **94** 154–166.

Lovasz, L. 1983. Submodular functions and convexity. A. Bachem, M. Grötschel, B. Korte, eds. *Mathematical Programming*: *The State of the Art*. Springer-Verlag, Berlin, Germany. 235–257.

Minoux, M. 1977. Accelerated greedy algorithms for maximizing submodular set functions. *Actes Congres IFIP*. J. Stoer, ed. Springer Verlag, Berlin, Germany. 234–243.

Nemhauser, G. L., L. A. Wolsey. 1981. Maximization submodular set functions: formulations and analysis of algorithms. P. Hansen, ed. *Studies on Graphs and Discrete Programming*. North-Holland Publishing, Amsterdam, The Netherlands 279–301.

——, ——, M. L. Fisher. 1978. An analysis of approximations for maximizing submodular set functions-I. *Math. Programming* **14** 265–294.

Nering, E. D., A. W. Tucker. 1993. *Linear Programs and Related Problems*. Academic Press, Inc., San Diego, CA.

Petrov, A., V. Cherenin. 1948. An improvement of train gathering plans design methods. (In Russian). *Zheleznodorozhnyi Trans.* **3** 60–71.

Robertazzi, T. G., S. C. Schwartz. 1989. An accelerated sequential algorithm for producing D-optimal designs. *SIAM J. Sci. Statist. Comput.* **10** 341–358.

Accepted by Thomas M. Liebling; received March 1998. *This paper has been with the authors* 3 *months for* 1 *revision.*