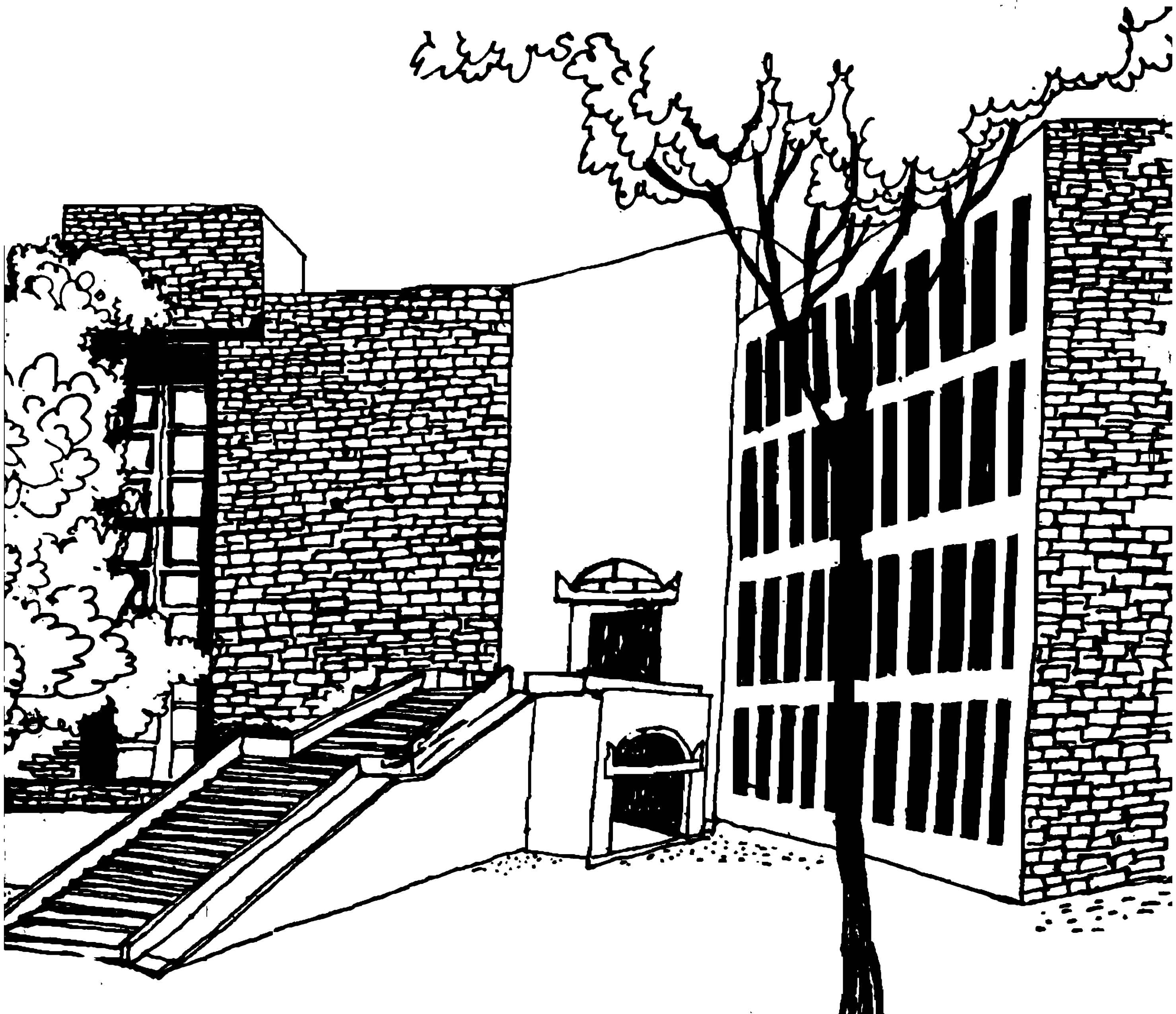
विद्याविनियोगाद्विकासः
IIM
AHMEDABAD

Working Paper



The Data-Correcting Algorithm for the Maximization of Submodular Functions: A Multilevel Search in the Hasse Diagram

Boris Goldengorin
Diptesh Ghosh

W.P.No. 2002-06-02
June 2002 1704

INDIAN INSTITUTE OF MANAGEMENT
AHMEDABAD-380 015
INDIA

11M11
WP-2002-06-02

THE DATA-CORRECTING ALGORITHM FOR THE MAXIMIZATION OF SUBMODULAR FUNCTIONS: A MULTILEVEL SEARCH IN THE HASSE DIAGRAM

BORIS GOLDENGORIN AND DIPTESH GHOSH

ABSTRACT. The Data-Correcting Algorithm (DCA) is a recursive branch and bound type algorithm, in which the data of a given problem instance is 'heuristically corrected' at each branching in such a way that the new instance will be as close as possible to polynomially solvable and the optimal solution to the corrected instance satisfies a prescribed accuracy condition (the difference between optimal and current solution). Recently (see [14]) we have applied the DCA to the minimization (maximization) a supermodular (submodular) function and can solve Quadratic Cost Partition Problem (QCP) instances on dense graphs with up to 100 vertices to optimality. In this paper we improve the above mentioned DCA for the submodular functions by searching the Hasse diagram more thoroughly at each subproblem. We study the behavior of the DCA with respect to the number of search levels of the Hasse diagram for the case of the QCP. Our computational experiments with QCP instances similar to those in [23] show that searching three levels of the Hasse diagram is an optimal strategy for QCP instances. Computational experiments with the improved DCA allow us to solve QCP instances on dense graphs with number of vertices up to 500 within 10 minutes on a standard personal computer.

Keywords: Data Correcting, Hasse Diagram, Multilevel Search, Quadratic Cost Partition Problem

1. INTRODUCTION

Many combinatorial optimization problems have as an underlying model the maximization of a submodular (or, equivalently, minimization of a supermodular) function, among them being the simple plant location (SPL) problem, generalized transportation problems, the Quadratic Cost Partition Problem (QCP) with nonnegative edge weights, set covering and other well known problems involving the minimization of pseudo-Boolean functions (see [26], [24], [2]).

Although the general problem of the maximization of a submodular 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 submodularity of the underlying objective function has been only implicit to the analysis. For example, [2] 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. [1] applies Lagrangean heuristics to several classes of location problems including SPL problems and reports results of extensive experiments on a Cray supercomputer. [23] have studied the quadratic cost partition problem (QCP) of which max-cut with nonnegative edge weights is a special case, again from the standpoint of polyhedral combinatorics. Recently [14] have applied the DCA to the minimization (maximization) a supermodular (submodular) function by which we can solve to optimality, for example, Quadratic Cost Partition Problem (QCP) instances on dense graphs up to 100 vertices. [10] reported their computational experiments for binary quadratic programs (BQP) with *adaptive memory tabu search* procedures. They assumed that the so called "c" instances with the number of vertices $n = 200$ and $n = 500$ "to be the most challenging problems reported in the literature to date - far beyond the capabilities of current exact methods and challenging as well for heuristic approaches". Since the BQP and QCP are equivalent (see, for example, [3], [16]) and Glover et al.'s BQP instances on dense graphs are defined, the "c" instances with $n = 200$ and $n = 500$ are 'statistically' equivalent to the corresponding QCP instances with $n = 200$ and $n = 500$

from Lee et al. To the best of our knowledge, the known algorithms solve to optimality the QCP instances on dense graphs with the number of vertices being at most 100. The purpose of this paper is to present an improved DCA which solves the QCP instances on dense graphs with number of vertices up to 500 within 10 minutes on a personal computer with 64MB RAM running at 300MHz.

The approach we take is to develop the class of *Data-Correcting Algorithms (DCA)*. This class of algorithms was introduced in [11] and [12] to solve 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. The polynomially solvable classes that we use in this paper are algorithmically determined.

For example, let us consider the DCA applied to an arbitrary function z defined on a set S and let y belongs to a polynomially solvable class $\mathcal{Y}$ of functions which is a subclass of a given class of functions $z \in Z$ defined also on S, and let $\rho(z,y) = \max\{|z(s) - y(s)| \ : s \in S\}$ be the *proximity measure*. We use the proximity measure $\rho(z,y)$ in the framework of the DCA for finding, by means of a heuristic procedure, an instance $y \in \mathcal{Y}$ that is as "close" as possible to z. 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 minimization of a supermodular function we have used the so called *Preliminary Preservation Algorithm (PPA)* for determining the relevant polynomial solvable class of supermodular functions (*PP-functions*, see [14]). In the following theorem, which is first published in [14] (see also examples of this theorem for the Simple Plant Location and Traveling Salesman Problems in [12]), it is formulated that the proximity measure is an upper bound for the difference of the optimum values of z and y on S, denoted by $z^*(S)$ and $y^*(S)$, respectively. Then the following holds: $|z^*(S) - y^*(S)| \leq \rho(z,y)$. If $\rho(z,y)$ is smaller than the value of prescribed accuracy ε_0, then the problem of finding $z^*(S)$ with the given prescribed accuracy ε_0 is solved by $y^*(S)$. Otherwise, the DCA decreases the current value of $\rho(z,y)$ by means of a branching procedure. More information about data-correcting algorithms (general scheme, comparison with branch and bound type algorithms, steps of construction, methods, etc.) can be found in [11] and [12].

The main difference between our previous algorithm [14] and the algorithm proposed in this paper is an extension of the used solution space more than one level deep in the Hasse diagram (see, for example, [15]). Our previous DCA [14], is based on the PPA at each step of which we use just two neighboring levels in the Hasse diagram. We call such PPA the *PPA of order zero*. The PPA of order zero is based on the following Theorem 1 (see [14]).

Theorem 1. *Let $[S,T] = \{I | S \subseteq I \subseteq T\}$ be the whole set of subsets defined by any pair of embedded sets S and T such that $S \subset T$ (the Hasse diagram is spanned on these sets S and T) and let z be a submodular function on $[S,T]$ with $k \in T \setminus S$. Then the following assertions hold.*
(a) $z^*[S+k,T] - z^*[S,T-k] \leq z(S+k) - z(S) = d_k^+(S)$.
(b) $z^*[S,T-k] - z^*[S+k,T] \leq z(T-k) - z(T) = d_k^-(T)$.

Here, $z^*[S,T] = \max\{z(I) : I \in [S,T]\}$. As is easy to see, the differences $d_k^+(S)$ and $d_k^-(T)$ are used just for two neighboring sets each of which can be obtained by adding to S (or deleting from T) a single element $k \in T \setminus S$. In [14] these differences are used for "correcting" the current data (values of z) in the DCA. In this paper we will use differences $z(P) - z(S)$ and $z(Q) - z(T)$ defined not only for the neighboring sets but also for any pair of embedded subsets, say $S \subset P$ and $Q \subset T$, such that $\max\{|P \setminus S|, |T \setminus Q|\} \leq r \leq |T \setminus S|$, and generalize the above mentioned assertions (see Theorems 6, 8, and 9 and Corollaries 7 and 10). Based on these new assertions we present a generalization of the PPA of order zero which is called the *PPA of order r*, and the corresponding DCA which is based on the PPA of order r (the DCA(PPAr)).

We have organized our paper as follows. In Section 2 we briefly review old results about the PPA (see Theorem 2 and Corollaries 3 and 4), which we use for determining the relevant polynomially solvable class of submodular functions (PP-functions). In Section 3 we present a new generalization of the PPA, called the PPA of order r (PPAr) (based on Theorems 6, 8, and 9 and Corollaries 7 and 10). We then describe a DCA based on the PPAr (the DCA(PPAr)) in Section 4. Sections 5 and 6 present the computational aspects of the paper. In Section 5 we briefly review different formulations of the Quadratic Cost Partition Problem (QCP) including the Quadratic Zero-One Programming statement of the QCP and the corresponding computer experiments with random QCP instances which are 'statistically' similar to those studied in [23]. Since the QCP and the Quadratic Zero-One Programming Problem (QZOP) are equivalent, i.e. for each QCP instance we can easily construct a corresponding QZOP instance (see e.g., [3] and [16]), our computational experiments with the QCP instances are comparable to the computational experiments with the QZOP instances. In Section 6 we solve hard and large instances of the QCP with DCA(PPAr) and demonstrate an improvement upon published results from [23] and [14], particularly when the data corresponds to dense graphs including instances of the QCP with 200 to 500 vertices (see, [10]). Section 7 concludes the paper.

2. A POLYNOMIALLY SOLVABLE CASE OF THE MAXIMIZATION OF A SUBMODULAR FUNCTION

In this section we introduce a polynomially solvable case for the maximization of submodular functions. In order to do this, we use the *Preliminary Preservation Algorithm* (PPA) (see [9]) which helps to construct the polynomially solvable case of submodular functions, called *PP-functions*. Details about the PPA and PP-functions can be found in [14] (see also [12]).

Let z be a real-valued function defined on the power set 2^N of $N = \{1, 2, \ldots, n\}$; $n \geq 1$. For each $S, T \in 2^N$ with $S \subseteq T$, define

$$[S,T] = \{I \in 2^N \mid S \subseteq I \subseteq T\}.$$

Note that $[\emptyset, N] = 2^N$. Any *interval* $[S,T]$ is a subinterval of $[\emptyset, N]$ if $\emptyset \subseteq S \subseteq T \subseteq N$. We denote this using the notation $[S,T] \subseteq [\emptyset, N]$. In this paper an interval is always a subinterval of $[\emptyset, N]$. Throughout this paper, it is assumed that z attains a finite maximum value on $[\emptyset, N]$ which is denoted by $z^*[\emptyset, N]$, and $z^*[S,T] = \max\{z(I) : I \in [S,T]\}$ for any $[S,T] \subseteq [\emptyset, N]$. The function z is called *submodular* on $[S,T]$ if for each $I, J \in [S,T]$ it holds that

$$z(I) + z(J) \geq z(I \cup J) + z(I \cap J).$$

Let us construct a polynomially solvable case of the maximization of a submodular function. We determine this case by the so called Preliminary Preservation Algorithm (PPA). The following theorem and corollary (from [14]) form the basis of the PPA.

Theorem 2. *Let z be a submodular function on $[S,T] \subseteq [\emptyset, N]$ and let $k \in T \backslash S$. Then the following assertions hold.*

(a) $z^*[S+k, T] - z^*[S, T-k] \leq z(S+k) - z(S)) = d_k^+(S)$.

(b) $z^*[S, T-k] - z^*[S+k, T] \leq z(T-k) - z(T) = d_k^-(T)$.

Corollary 3. *(Preservation rules of order zero). Let z be a submodular function on $[S,T] \subseteq [\emptyset, N]$ and let $k \in T \backslash S$. Then the following assertions hold.*

First Preservation Rule (FPR)

(a) *If $d_k^+(S) \leq 0$, then $z^*[S,T] = z^*[S, T-k] \geq z^*[S+k, T]$.*

Second Preservation Rule (SPR)

(b) *If $d_k^-(T) \leq 0$, then $z^*[S,T] = z^*[S+k, T] \geq z^*[S, T-k]$.*

The following corollary extends the rules in Corollary 3.

Corollary 4. *Let z be a submodular function on $[S,T] \subseteq [\emptyset, N]$ and let $k_i \in T \backslash S$, $i \in R = \{1, \ldots, r\}$. Then the following assertions hold.*

First Extended Preservation Rule

(a) *If $d_{k_i}^+(S) \leq 0$ for all $i \in R$, then $z^*[S,T] = z^*[S, T \backslash (\cup_{i \in R} k_i)]$.*

Second Extended Preservation Rule

(b) *If $d_{k_i}^-(T) \leq 0$ for all $i \in R$, then $z^*[S,T] = z^*[S \cup (\cup_{i \in R} k_i), T]$.*

Proof. We prove only part (a) since the proof of (b) is similar. Applying the FPR does not change the values $d_{k_i}^+(S)$ since only the set T changes and not the set S. Let us apply the FPR with $d_{k_1}^+(S) \leq 0$, then the interval $[S.T]$ will be reduced to the interval $[S, T - k_1]$. The values $d_{k_i}^+(S)$,

$i \in R - k_1$ have not changed for the new interval $[S, T - k_1]$ and hence after applying the FPR with $d^+_{k_2}(S) \leq 0$, then the interval $[S, T - k_1]$ will be reduced to the interval $[S, (T - k_1) - k_2]$. Repeating the application of the FPR $(r - 2)$ times completed the proof. ■

Based on Corollary 4 it is often possible to exclude a large part of $[\emptyset, N]$ from consideration when determining a global maximum of z on $[\emptyset, N]$. The PPA determines a subinterval $[S, T]$ of $[\emptyset, N]$ that certainly contains a global maximum of z, whereas $[S, T]$ cannot be made smaller by using the preservation rules of Corollary 4. In case the PPA stops with $S = T$ then an optimal solution has been found, i.e. $S \in argz^*[\emptyset, N]$.

```
          The Preliminary Preservation Algorithm
Procedure PPA($X, Y; S, T$)
Input:    A submodular function $z$ on interval $[X, Y]$ of $[\emptyset, N]$
Output:   The subinterval $[S, T] \subseteq [X, Y]$ such that $z^*[S, T] =$
          $z^*[X, Y]$ and $\min\{d^+_i(S), d^-_i(T) \mid i \in T \backslash S\} > 0$.
Step 0:   $S \leftarrow X$, $T \leftarrow Y$, $S_1 \leftarrow \emptyset$, $T_1 \leftarrow \emptyset$.
Step 1:   $S_1 \leftarrow \{ k \in T \backslash S \mid d^-_k(T) \leq 0\}$, $S \leftarrow S \cup S_1$.
Step 2:   $T_1 \leftarrow \{ k \in T \backslash S \mid d^+_k(S) \leq 0\}$, $T \leftarrow T \backslash T_1$.
Step 3:   If $S = T$, then end.
Step 4:   If $T_1 \neq \emptyset$, then go to Step 1.
Step 5:   If $S_1 \neq \emptyset$, then go to Step 2, else end.
```

Every time the interval $[S, T]$ is updated the conditions of Corollary 4 are satisfied. Each new interval contains a global maximum because at each step $z^*[S, T] = z^*[X, Y]$. Since the PPA continues the calculation if and only if at each stage, at least one of S and T is updated, the correctness of the PPA is clear. In [12] it is proved that the time complexity of PPA is $O(n^2)$.

Any submodular function z on $[X, Y]$ for which the PPA returns a global maximum for z is called a *PP-function*. Theorem 5 states an interesting property of PP-functions in terms of strict component of local maxima.

A subset $L \in [\emptyset, N]$ is called a *local maximum* of z if for each $i \in N$

$$z(L) \geq \max\{z(L - i), z(L + i)\}.$$

A subset $S \in [\emptyset, N]$ is called a *global maximum* of z if $z(S) \geq z(I)$ for each $I \in [\emptyset, N]$. We will use the Hasse diagram (see e.g., [15]) as the ground graph $G = (V, E)$ in which $V = [\emptyset, N]$ and a pair (I, J) is an edge if and only if either $I \subset J$ or $J \subset I$, and $|I \setminus J| + |J \setminus I| = 1$. The graph $G = (V, E)$ is called *z-weighted* if the weight of each vertex $I \in V$ is equal to $z(I)$; and is denoted by $G = (V, E, z)$. A local maximum $\underline{L} \in [\emptyset, N]$ ($\overline{L} \in [\emptyset, N]$) is called a *lower* (respectively, *upper*) *maximum* if there is no another local maximum L such that $L \subset \underline{L}$ (respectively, $\overline{L} \subset L$).

If an interval $[\underline{L}, \overline{L}]$ with $\underline{L} \subseteq \overline{L}$ has lower and upper maxima as its end points, then the submodular function is constant over this interval. We can use such intervals to obtain a representation of connected subgraphs of local maxima.

Let V_0 be the subset of V corresponding to all local maxima of z and let $H_0 = (V_0, E_0, z)$ be the subgraph of G induced by V_0. Note that this subgraph consists of at least one connected component. We denote the connected components by $H_0^j = (V_0^j, E_0^j, z)$, with $j \in J_0 = \{1, \ldots, r\}$. Note that if L_1 and L_2 are vertices in the same component then $z(L_1) = z(L_2)$.

A component H_0^j is called a *strict local maximum component* (STC) if for each $I \notin V_0^j$, for which there is an edge (I, L) with $L \in V_0^j$, we have $z(I) < z(L)$. A component H_0^j is called a *saddle local maximum component* (SDC) if for some $I \notin V_0^j$, there exists an edge (I, L) with $L \in V_0^j$ such that $z(I) = z(L)$.

[20] has observed that any global maximum belongs to an STC. The following Theorem 5 can be found in [12] and [14].

Theorem 5. *If z is a submodular PP-function on $[X, Y] \subseteq [\emptyset, N]$, then $[X, Y]$ contains exactly one STC.*

Note that not each submodular function with exactly one STC on $[\emptyset, N]$ is a PP-function. For example, let $N = \{1, 2, 3\}$ and consider the submodular function z defined by $z(I) = 2$ for any

$I \in [\emptyset, \{1,2,3\}] \setminus (\{\emptyset\} \cup \{1,2,3\})$ and $z(I) = 1$ for $I \in (\{\emptyset\} \cup \{1,2,3\})$. The vertex set of the unique STC defined by this function can be represented by $[\{1\},\{1,2\}] \cup [\{1\},\{1,3\}] \cup [\{2\},\{1,2\}] \cup [\{2\},\{2,3\}] \cup [\{3\},\{1,3\}] \cup [\{3\},\{2,3\}]$. The PPA terminates with $[S,T] = [\emptyset, \{1,2,3\}]$ and so, z is not a PP-function.

In case of the QCP, the PPA does not decrease the interval $[S,T]$ if $d_k^+(S) = p_k - \sum_{i\in S} q_{ik} > 0$ and $d_k^-(T) = \sum_{i \in T} q_{ik} - p_k > 0$. If there is a sequence such that $\emptyset \subset S_1 \subset S_2 \subset \cdots \subset S_p = T_{n-p} \subset T_{n-p+1} \subset \cdots \subset T_{n-1} \subset N$ with $d_k^+(S_i) \le 0$ and $d_k^-(T_{n-i}) \le 0$ for all $i = 1, \dots, p$ and some $p \le |T \setminus S|$ the PPA solves the QCP. Therefore the corresponding class of submodular functions can be expound as PP-functions.

In the next section we determine a generalization of the PPA, called the *PPA of order r* (PPAr).

3. PPA of order r

The preservation rules in the PPA examine solutions that are only one level deeper in the Hasse diagram than the current solution. The following statements allow us to explore a larger portion of the solution space. This may be useful because we obtain additional possibilities for narrowing the original interval (see [13]).

Theorem 6. *Let z be a submodular function on $[S,T] \subseteq [\emptyset, N]$ and let $k \in T\setminus S$. Then the following assertions hold.*

(a) For any $t_0^+(k) \in \arg\max\{d_k^+(S+t) : t \in T\setminus(S+k)\}$,
$z^*[S+k,T] - \max\{z^*[S,T-k], z(S+k)\} \le \max\{d_k^+(S+t_0^+(k)), 0\}$.

(b) For any $t_0^-(k) \in \arg\max\{d_k^-(T-t) : t \in T\setminus(S+k)\}$,
$z^*[S,T-k] - \max\{z^*[S+k,T], z(T-k)\} \le \max\{d_k^-(T-t_0^-(k)), 0\}$.

Proof. We prove only part (a) since the proof of (b) is similar. Let

$$t_1^+(k) \in \arg\max\{z^*[S+k+t,T] : t \in T\setminus(S+k)\}$$

We may represent the partition of $[S,T]$ by means of its subintervals as follows:

$$[S,T] = S \cup \bigcup_{t \in T\setminus S} [S+t, T].$$

Applying this representation on the interval $[S+k,T]$ we have

$$z^*[S+k,T] = \max\{z(S+k), z^*[S+k+t_1^+(k),T]\}.$$

We distinguish now the following two cases:

Case 1: $z(S+k) \le z^*[S+k+t_1^+(k),T]$. Then $z^*[S+k,T] = z^*[S+k+t_1^+(k),T]$. For any $k \in T\setminus[S+t_1^+(k)]$ Theorem 2(a) on the interval $[S+t_1^+(k),T]$ states that:

$$z^*[S+t_1^+(k)+k,T] - z^*[S+t_1^+(k),T-k] \le d_k^+(S+t_1^+(k)),$$

i.e., after substituting $z^*[S+k,T]$ instead of $z^*[S+k+t_1^+(k),T]$ this inequality can be written as follows:

$$z^*[S+k,T] - z^*[S+t_1^+(k),T-k] \le d_k^+(S+t_1^+(k)),$$

and taking into account that $z^*[S+t_1^+(k),T-k] \le z^*[S,T-k]$ we have

$$z^*[S+k,T] - z^*[S,T-k] \le d_k^+(S+t_1^+(k)).$$

Adding two maximum operations leads to the following inequality

$$z^*[S+k,T] - \max\{z^*[S,T-k], z(S+k)\} \le \max\{d_k^+(S+t_1^+(k)),0\}.$$

Finally, $d_k^+(S+t_1^+(k)) \le d_k^+(S+t_0^+(k))$ since $d_k^+(S+t)$ was maximal for $t_0^+(k)$. This gives the required result.

Case 2: $z(S+k) > z^*[S+k+t_1^+(k),T]$.

Then $z^*[S+k,T] = z(S+k)$. Consider the inequality

$$z(S+k) - \max\{z^*[S,T-k], z(S+k)\} \leq 0.$$

Since $z(S+k) = z^*[S+k,T]$ we have

$$z^*[S+k,T] - \max\{z^*[S,T-k], z(S+k)\} \leq 0.$$

Adding a maximum operation with $d_k^+(S+t_0^+(k))$ gives the required result

$$z^*[S+k,T] - \max\{z^*[S,T-k], z(S+k)\} \leq \max\{d_k^+(S+t_0^+(k)), 0\}.$$

■

Corollary 7. *(Preservation rules of order one). Let z be a submodular function on $[S,T] \subseteq [\emptyset, N]$ and let $k \in T\backslash S$. Then the following assertions hold.*

First Preservation Rule of Order One

(a) If $\max\{d_k^+(S+t) : t \in T\backslash(S+k)\} \leq 0$, then
$z^*[S,T] = \max\{z^*[S,T-k], z(S+k)\} \geq z^*[S+k,T]$

Second Preservation Rule of Order One

(b) If $\max\{d_k^-(T) : t \in T\backslash(S+k)\} \leq 0$, then
$z^*[S,T] = \max\{z^*[S+k,T], z(T-k)\} \geq z^*[S,T-k]$

If the current interval $[S,T]$ cannot be narrowed by preservation rules of order one then the same interval cannot be narrowed by preservation rules of order zero (defined in Corollary 3). Moreover, if the interval $[S,T]$ can be narrowed by preservation rules of order zero then this interval can be narrowed by preservation rules of order one. We prove this in the following theorem.

Theorem 8. *Preservation rules of order one are not weaker than preservation rules of order zero.*

Proof. We compare only first preservation rules of order one and order zero because the proof for case of second preservation rules is similar.

Assume that the preservation rule of order one is not applicable, i.e., $\max\{d_k^+(S+t) : t \in T\backslash(S+k)\} = d_k^+(S+t_0) > 0$. The definition of submodularity of z implies $d_k^+(S) \geq d_k^+(S+t_0)$. Hence, $d_k^+(S) > 0$ and the first preservation rule is not applicable. In case when the first preservation rule of order zero is applicable, i.e., $d_k^+(S) \leq 0$ we have $0 \geq d_k^+(S) \geq d_k^+(S+t)$ for all $t \in T\backslash(S+k)$, i.e., $\max\{d_k^+(S+t) : t \in T\backslash(S+k)\} \leq 0$ which leads to a contradiction. ■

Note that the computational complexity for rules of order one and order zero is different not only in their time complexities but also in their space complexities. This is because, together with the preserved interval either $[S+k,T]$ or $[S,T-k]$ we should preserve exactly one additional value either $z(T-k)$ or $z(S+k)$, respectively. This property is also valid for preservation rules of order $r \geq 1$.

Instead of looking one level deep we may look r levels deep in order to determine whether we can include or exclude an element. To simplify the presentation of the following theorem, we need some new notations describing certain subsets of the interval $[S,T]$. Let

$M_r^+[S,T] = \{I \in [S,T] : |I\backslash S| \leq r\}$,
$M_r^-[S,T] = \{I \in [S,T] : |T\backslash I| \leq r\}$.

The set $M_r^+[S,T]$ is a collection of all sets representing solutions containing more elements than S, and which are no more than r levels deeper than S in the Hasse diagram. Similarly, the set $M_r^-[S,T]$ is a collection of all sets representing solutions containing less elements than T, and which are no more than r levels deeper than T in the Hasse diagram. Define further the collections of sets

$N_r^+[S,T] = M_r^+[S,T]\backslash M_{r-1}^+[S,T]$,
$N_r^-[S,T] = M_r^-[S,T]\backslash M_{r-1}^-[S,T]$

The sets $N_r^+[S,T]$ and $N_r^-[S,T]$ are the collection of sets which are located on the level r above S and below T in the Hasse diagram, respectively. Let $v_r^+[S,T] = \max\{z(I) : I \in M_r^+[S,T]\}$, $v_r^-[S,T] = \max\{z(I) : I \in M_r^-[S,T]\}$, $w_{rk}^+[S,T] = \max\{d_k^+(I) : I \in N_r^+[S+k,T]\}$ and $w_{rk}^-[S,T] = \max\{d_k^-(I) : I \in N_r^-[S,T-k]\}$.

Theorem 9. *Let z be a submodular function on $[S,T] \subseteq [\emptyset, N]$ with $k \in T\backslash S$ and let r is a positive integer. Then the following assertions hold.*

(a) If $|N_r^+[S+k,T]| > 0$, then
$z^*[S+k,T] - \max\{z^*[S,T-k], v_r^+[S,T]\} \leq \max\{w_{rk}^+[S,T], 0\}$.

(b) If $|N_r^-[S,T-k]| > 0$, then
$z^*[S,T-k] - \max\{z^*[S+k,T], v_r^-[S,T]\} \leq \max\{w_{rk}^-[S,T], 0\}$.

Proof. We prove only part (a) since the proof of the part (b) is similar. We may represent the partition of interval $[S,T]$ as follows:

$$[S,T] = M_r^+[S,T] \cup \bigcup_{I \in N_r^+[S,T]} [I,T].$$

Applying this representation on the interval $[S+k,T]$ we have

$$z^*[S+k,T] = \max\{v_r^+[S+k,T], \max\{z^*[I+k,T] : I \in N_r^+[S,T]\}\}.$$

Let $I(k) \in \arg\max\{z^*[I+k,T] : I \in N_r^+[S,T]\}$, and let us consider two cases of the last equality:
Case 1. $z^*[I(k)+k,T] \geq v_r^+[S+k,T]$, and
Case 2. $z^*[I(k)+k,T] < v_r^+[S+k,T]$.

In the first case $z^*[S+k,T] = z^*[I(k)+k,T]$. For $I(k) \in N_r^+[S,T]$ Theorem 2(a) on the interval $[I(k),T]$ states:

$$z^*[I(k)+k,T] - z^*[I(k),T-k] \leq d_k^+(I(k)),$$

i.e. in case 1

$$z^*[S+k,T] - z^*[I(k),T-k] \leq d_k^+(I(k)).$$

Note for $[I(k),T-k] \subseteq [S,T-k]$ we have $z^*[S,T-k] \geq z^*[I(k),T-k]$. This leads to the following inequality

$$z^*[S+k,T] - z^*[S,T-k] \leq d_k^+(I(k)).$$

Adding two maximum operations gives

$$z^*[S+k,T] - \max\{z^*[S,T-k], v_r^+[S+k,T]\} \leq \max\{d_k^+(I(k)), 0\}.$$

Since $w_{rk}^+[S,T]$ is the maximum of $d_k^+(I)$ for $I \in N_r^+[S+k,T]$, we have the required result.

In the second case $z^*[S+k,T] = v_r^+[S+k,T]$ the following equality holds:

$$z^*[S+k,T] - v_r^+[S+k,T]\} = 0$$

or

$$z^*[S+k,T] - \max\{z^*[S,T-k], v_r^+[S+k,T]\} \leq 0.$$

Adding a maximum operation with $w_{rk}^+[S,T]$ completes the proof of case (a)

$$z^*[S+k,T] - \max\{z^*[S,T-k], v_r^+[S+k,T]\} \leq \max\{w_{rk}^+[S,T], 0\}.$$

∎

Corollary 10. *(preservation rules of order r). Let z be a submodular function on $[S,T] \subseteq [\emptyset, N]$ and let $k \in T\backslash S$. Then the following assertions hold.*

First Preservation Rule of Order r

(a) If $w_{rk}^+[S,T] \leq 0$, then
$z^*[S,T] = \max\{z^*[S,T-k], v_r^+[S+k,T]\} \geq z^*[S+k,T]$

Second Preservation Rule of Order r

(b) If $w_{rk}^-[S,T] \leq 0$, then
$z^*[S,T] = \max\{z^*[S+k,T], v_r^-[S,T-k]\} \geq z^*[S,T-k]$

Note that the analogue of Theorem 8 can be proved for preservation rules of order $r-1$ and r by induction.

Now we can describe the PPA of order r (PPAr). The PPAr behaves in the same manner as the PPA, i.e., it tries to decrease the original interval $[X, Y]$ in which an optimal solution is located. The difference between the two algorithms lies in the fact that the PPA searches only one level deep in the Hasse diagram, while the PPAr searches r levels deep. The PPAr chooses one element to investigate further from either the top or the bottom of the Hasse diagram. We could investigate all vertices from $T \setminus S$ but this would cost too much time. Therefore we use a heuristic to select the element which we investigate further. The element we choose is an element for which it is likely that one of the preservation rules of order r will succeed in including or excluding this element from an optimal solution. The preservation rules of order one apply if $\max\{d_k^+(S+t) : t \in T \setminus (S+k)\} \leq 0$ or $\max\{d_k^-(T-t) : t \in T \setminus (S+k)\} \leq 0$. So if we want them to apply then we have to choose an element k so as to minimize the values $d_k^+(S+t)$ and $d_k^-(T-t)$. According to an equivalent definition of a submodular function (see [26]), $d_k^+(S) \geq d_k^+(S+t)$, if we choose $d_k^+(S)$ as small as possible, then $d_k^+(S+t)$ will not be large and hopefully negative for all t, and the first preservation rule of order one is more likely to apply. Also if we take k with the smallest value $d_k^-(T)$ then the second preservation rule of order one is more likely to apply. Our computational study (see Section 5) selects the best value of r, and therefore shows the relevance of this choice.

It is clear that if we search deep enough, the PPAr will always find an optimal solution to our problem. We just take $r = |Y \backslash X|$, where $[X, Y]$ is the initial interval, and at each step we will be able to include or exclude an element of the initial interval. However, the number of sets we have to examine in this case is not a polynomial function of r.

Let us define two recursive procedures PPArplus and PPArminus by means of which we can try to include and exclude some elements of the initial interval $[X, Y]$.

```
Procedure PPArplus($S, T, k, r, maxd$)
begin
        Calculate $z(S+k)$. If $z(B) < z(S+k)$ then $B \leftarrow S+k$;
        For all $t \in T\backslash(S+k)$ calculate $d_k^+(S+t)$.
        If $d_k^+(S+t) \leq 0$ or $r = 1$ then $maxd \leftarrow \max\{maxd, d_k^+(S+t)\}$
        else call PPArplus($S+t, T, k, r-1, maxd$)
end
Procedure PPArminus($S, T, k, r, maxd$)
begin
        Calculate $z(T-k)$. If $z(B) < z(T-k)$ then $B \leftarrow T-k$;
        For all $t \in T\backslash(S+k)$ calculate $d_k^-(T-t)$.
        If $d_k^-(T-t) \leq 0$ or $r = 1$ then $maxd \leftarrow \max\{maxd, d_k^-(T-t)\}$
        else call PPArplus($S, T-t, k, r-1, maxd$)
end
```

```
The Preliminary Preservation Algorithm of order $r$
Input:   A submodular function $z$ on $[X, Y]$ of $[\emptyset, N]$
Output:  The subinterval $[S, T]$ and the set $B$ such that
         $z^*[X, Y] = \max\{z^*[S, T], z(B)\}$ and
         $\min\{w_{rk}^+[S, T], w_{rk}^-[S, T]\} > 0$ for all $k \in T\backslash S$

Step 0:  $S \leftarrow X, T \leftarrow Y, B \leftarrow \emptyset$.

Step 1:  call $PPA(X, Y; S, T)$; go to Step 2;

Step 2:  $d^+ \leftarrow \max\{d_k^+(S) : k \in T\backslash S\}$, $d^- \leftarrow \max\{d_k^-(T) : k \in T\backslash S\}$;
         If $d^+ < d^-$ then go to Step 3 else go to Step 4.
```

```
Step 3:  $k \leftarrow \arg\max\{d_t^+(S) : t \in T\backslash S\}$;
         call $PPArplus(S, T, k, r, maxd)$;
         If $maxd \leq 0$ then $T \leftarrow T - k$, go to Step 1 else end.
Step 4:  $k \leftarrow \arg\max\{d_t^-(T) : t \in T \setminus S\}$;
         call $PPArminus(S, T, k, r, maxd)$;
         If $maxd \leq 0$ then $S \leftarrow S + k$, go to Step 1 else end.
```

Note that the PPAr finds a maximum of the submodular function iff the level r of the Hasse diagram is "deeper or equal" to the level on which a STC is located.

4. The Data-Correcting Algorithm (DCA)

In this section we briefly describe the main idea and the structure of the DCA based on the PPAr. We will call this DCA the DCA with PPAr and abbreviate to DCA(PPAr). The description of the DCA(PPA) can be found in [14]. In this section we will point out the main differences between the DCA(PPA) and the DCA(PPAr).

Recall that if a submodular function z is not a PP-function, then the PPA terminates with a subinterval $[S, T]$ of $[\emptyset, N]$ with $S \neq T$ containing a maximum of z without knowing its exact location in $[S, T]$. In this case, the post-condition $\min\{d_i^+(S), d_i^-(T) \mid i \in T\backslash S\} = \delta > 0$ of the PPA is satisfied. The basic idea of the DCA is that if a situation occurs for which this post-condition holds, then the data of the current problem will be corrected in such a way that a corrected function z violates at least one of inequalities $d_k^+(S) = \delta > 0$ or $d_p^-(T) = \delta > 0$ for some $k, p \in T\backslash S$. In that manner the PPA can continue. Moreover, each correction of z is carried out in such a way that the new (corrected) function remains submodular. If the PPA stops again without an optimal solution to the corrected instance, we apply the correcting rules again and so on until the PPA finds an optimal solution. For $k \in T\backslash S$ Theorem 11 gives upper bounds for the values $z^*[S, T] - z^*[S, T-k]$, namely, $d_k^+(S)$, and for $z^*[S, T] - z^*[S+k, T]$, namely, $d_k^-(T)$. So, if we choose to include an element k in the interval $[S, T]$, then we know that the difference between an optimal solution of the original interval and the new one will be smaller than $d_k^+(S)$. A similar interpretation holds for $d_k^-(T)$. It is clear that after at most n corrections we will find an approximate solution $J \in [\emptyset, N]$ such that $z^*[\emptyset, N] \leq z(J) + \varepsilon$, where $\varepsilon = \sum_{i=1}^n \delta_i$ with δ_i equal to either $d_i^+(S)$ or $d_i^-(T)$. These arguments lead to the following two upper bounds (see [21], [25]).

Theorem 11. *If* $\min\{d_i^+(S), d_i^-(T) \mid i \in T\backslash S\} > 0$, *then*

$$ub_1 = z(S) + \sum_{i \in T\backslash S} d_i^+(S) \geq z^*[S, T]$$

and

$$ub_2 = z(T) + \sum_{i \in T\backslash S} d_i^-(T) \geq z^*[S, T].$$

Before the PPA stops there are a few options. First, if we would like to allow a certain prescribed accuracy, say ε_0, of an approximate solution for the current interval $[S, T]$, then after each correction we must check the inequalities $z^*[X, Y] - z^*[S, T] \leq \varepsilon \leq \varepsilon_0$. If $\varepsilon > \varepsilon_0$ then it is possible to look deeper than one level in the Hasse diagram (see the PPAr) either to determine whether or not an element belongs to an optimal solution or at least to reduce the current values of $d_i^+(S)$ and $d_i^-(T)$, because $w_{rk}^+[S, T] \geq w_{r+1k}^+[S, T]$ and $w_{rk}^+[S, T] \geq w_{rk}^+[S, T-t]$, or $w_{rk}^-[S, T] \geq w_{r+1k}^-[S, T]$ and $w_{rk}^-[S, T] \geq w_{rk}^-[S-t, T]$. We will explore these possibilities in the DCA(PPAr). Finally, we can divide the current problem into two subproblems by splitting the corresponding interval into $[S+k, T]$ and $[S, T-k]$ for some chosen k, and apply the PPA on each interval separately. The monotonicity property $d_i^+(S) \geq d_i^+(S+t)$ of a submodular function is the base of the following branching rule (see [9]).

Branching Rule. For $k \in \arg\max\{\max[d_i^+(S), d_i^-(T)] : i \in T\backslash S\}$, split the interval $[S, T]$ into two subintervals $[S+k, T]$, $[S, T-k]$, and use the prescribed accuracy ε of $[S, T]$ for both intervals.

To make the DCA more efficient we incorporate improved upper bounds by which we can discard certain subproblems from further consideration. We may discard a subproblem if some optimal value found so far is larger than the upper bound of the subproblem under investigation because the optimal value of this subproblem will never be larger then the optimal value found so far.

Using results due to [21], the upper bounds ub_1 and ub_2 from Theorem 11 can be tightened. Let us define the following sets of positive numbers: $d^+(S,T) = \{d_i^+(S) : d_i^+(S) > 0, i \in T\backslash S\}$ and $d^-(S,T) = \{d_i^-(T) : d_i^-(T) > 0, i \in T\backslash S\}$. let us also define the ordered arrays: $d^+[i]$ is an i-th largest element of $d^+(S,T)$ and $d^-[i]$ is an i-th largest element of $d^-(S,T)$ both for $i = 1,\ldots,|T\backslash S|$. So, $d^+[1] \geq \cdots \geq d^+[|T\backslash S|]$ and $d^-[1] \geq \cdots \geq d^-[|T\backslash S|]$. Let $z^*[S,T,i] = \max\{z(I) : N_i^+[S,T]\}$ which is the optimal value of $z(I)$. Finally, let us consider two functions which describe the behavior of our upper bounds while we add elements to the set S or delete elements from the set T: $f^+(i) = z(S) + \sum_{j=1}^{i} d^+[j]$ and $f^-(i) = z(T) + \sum_{j=1}^{i} d^-[j]$. Hence, $z^*[S,T,i] \leq \min\{f^+(i), f^-(i)\}$. Since $z^*[S,T] = \max\{z^*[S,T,i] : i = 1,\ldots,|T\backslash S|\}$ we have the following upper bound

$$ub = \max\{\min[f^+(i), f^-(i)] : i = 1,\ldots,|T\backslash S|\} \geq z^*[S,T].$$

Now we will describe the DCA. The DCA starts with a submodular function z on the interval $[\emptyset, N]$ and the prescribed accuracy ε_0 . A *list of unsolved subproblems* (LUS) is kept during the course of the DCA. Every time a subproblem is further decomposed into smaller subproblems, one of the subproblems is added to the LUS and on the other one the DCA is applied. After a solution has been found to a subproblem, a new subproblem is taken from the LUS, and so on until the LUS is empty. First, the DCA approximates a subproblem by using the PPA. If this does not result in an optimal solution of that subproblem, it first tries to discard the subproblem by using the upper bound, else the subproblem will be either corrected (if $\varepsilon \leq \varepsilon_0$) or (if $\varepsilon > \varepsilon_0$) split up by means of the branching rule.

Note that the corrections are executed implicitly. A correction allows the PPA to continue at least one step since the correction makes the post-condition of the PPA is invalid. For instance, if the PPA stops with an interval $[S,T]$, then after increasing (correction) the value of z on $[S, T-k]$ by $d_k^+(S) > 0$ the DCA may discard the subinterval $[S+k, T]$, because $z^*[S+k,T] - [z^*[S,T-k] + d_k^+(S)] \leq 0$. In fact, instead of correcting the function values of the preserved subinterval, the DCA increases the current value of ε with $d_k^+(S)$. In our example, if the value of the current accuracy of the interval $[S,T]$ is equal to ε, then after discarding the subinterval $[S+k,T]$ its value will be equal to $\varepsilon + d_k^+(S)$. These arguments show that the DCA does not change our submodular function explicitly. On the other hand, let $I \in [S+k,T]$, $J \in [S,T-k]$, then the submodularity of z implies $z(I) + z(J) \geq z(I \cap J) + z(I \cup J)$. Since $I \cap J \in [S, T-k]$ and $I \cup J \in [S+k, T]$, we have $z(I) + [z(J) + d_k^+(S)] \geq [z(I \cap J) + d_k^+(S)] + z(I \cup J)$. Therefore, by correcting the values of z on a subinterval, the DCA preserves the submodularity of z. Finally, note that using the PPAr instead of the PPA yields two advantages: either by improving the narrowing of the current interval or by decreasing the current value of ε.

5. Computational experiments with the QCP and QZOP: A brief review of the literature

The QCP can be described as follows (see e.g., [23]). Given nonnegative real numbers q_{ij} and real numbers p_i with $i,j \in N = \{1,2,\ldots,n\}$, the QCP is the problem of finding a subset $S \subset N$ such that the function

$$z(S) = \sum_{j\in S} p_i - \frac{1}{2}\sum_{i,j\in S} q_{ij}$$

will be maximized. The Max-Cut Problem (MCP) is a special case of the QCP, and can be described as follows. Consider an edge weighted undirected graph $U(N,E)$ with edge weights $w_{ij} \geq 0$, $ij \in E$. Define a cut $\delta(T)$ as the edge set containing all the edges with one end in T and the other end in $N \setminus T$. Define further the weight of a cut as the sum of the edge weights in the cut. The MCP is the problem of finding a cut, and thus a partition, with maximum possible weight. The MCP is a special case of the QCP, namely take

$$p_i = \sum_{i \in V} w_{ij} \text{ and } q_{ij} = 2w_{ij}.$$

An instance of the QCP is defined by an integer positive n, a vector of real numbers P_i, $i = 1, ..., n$, and a symmetric matrix $Q = ||q_{ij}||$, $i, j = 1, ..., n$ with nonnegative entries.

The QCP and the MCP arise in many real world applications (see [22] and references within) such as capital budgeting, time tabling, communication scheduling, statistical physics, design of printed circuit boards and VLSI design (see also [3], [6] and [23]). Since the MCP is a special case of the QCP, the QCP is also NP-hard [19]. An α-approximation algorithm is a polynomial-time algorithm that always finds a feasible solution with objective function value within a factor of α of optimal ([32]). The best known α-approximation algorithm for MCP gives $\alpha = 0.87856$ ([32]). On the negative side though, [17] has shown that there can be no 0.941-approximation algorithm for MCP unless $P = NP$. In other words, to solve the MCP with prescribed accuracy within 5.9% is an NP-hard problem.

The earliest formulation of the QCP (see [16]) in terms of an unconstrained quadratic zero-one programming problem (QZOP) is the following pseudo-Boolean formulation:

$$\max(\sum_{i=1}^{n} p_i x_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij} x_i x_j \mid x \in \{0,1\}^n).$$

Since $x_i^2 = x_i$ we can assume that the diagonal of $Q = ||q_{ij}||$ is zero.

The equivalence between QZOP and the MCP has been pointed out in [16] (see also [3]). Since the MCP is a special case of the QCP, the QZOP and the QCP are equivalent also. It means that in quadratic time for each instance of the QZOP we can find an instance of the QCP such that they have the same sets of feasible solutions and the same values of densities. This justifies the comparability of our computational experiments with the QCP instances and either QZOP or MCP instances reviewed below through the values of densities of the corresponding instances.

A mixed-integer programming (MIP) formulation of the QCP can be found in [27]. In this formulation the quadratic term is replaced by a linear one and a number of linear constraints:

$$\max(\sum_{i=1}^{n} p_i x_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij} y_{ij} \mid x_i + x_j - y_{ij} \leq 1;$$

$$\text{for } i, j = 1, \ldots n;\ x \in \{0,1\}^n,\ y \in \{0,1\}^{n \times n}).$$

Another MIP formulation is given in [23]:

$$\max(\sum_{i=1}^{n} p_i x_i - \lambda \mid \lambda \geq \sum_{ij \in E(T) \cup \delta(T)} q_{ij}(x_i + x_j - 1)$$

$$\text{for } T \subseteq N; x \in \{0,1\}^n, \lambda \geq 0)$$

where

$$E(T) = \{(i,j) \mid i \in T,\ j \in T,\ q_{ij} > 0\}$$

and

$$\delta(T) = \{(i,j) \mid i \in T,\ j \in N \backslash T,\ q_{ij} > 0\}.$$

An advantage of the latter formulation over the former is that it uses a smaller number of variables, although it requires an exponential number of constraints. The exponential number of constraints makes it impossible to solve the full formulation for large instances. In order to overcome this difficulty [23] start with only a small subset of the vertex set constraints ($\lambda \geq \sum_{ij \in E(T) \cup \delta(T)} q_{ij}(x_i + x_j - 1)$) and generate violated ones whenever they are needed. Therefore they need to solve the problem of recognizing violated constraints, i.e. separation problem for the vertex set constraints in their branch and cut algorithm. The separation problem is a typical part of branch and cut algorithms based on the polyhedral approach in combinatorial optimization. [5] have shown that the corresponding separation problems are polynomially solvable for a wide class of QZOPs.

The methods of computational studies of the QZOP can be classified into the following groups (see [22]): branch and bound methods [4], [28], linear programming based methods (branch and cut algorithms [3], [23]), eigenvalue based methods, and approaches via semidefinite programming [29], [30]. We will not discuss all of these approaches but restrict ourselves to one important remark. We have not found any computational study of *exact optimal solutions* for the QCP or QZOP for *dense* graphs in which the number of vertices is at least 60. An exception is a specialized exact algorithm for the maximum clique problem (which we consider as a special case of the general QZOP (the stability number problem in the complementary graph) in [7]. They give computational results for problems growing up to 100 variables with any edge density. However, the maximum clique problem is a special case of the QZOP. [4] presented a comprehensive analysis of computational results published in [3], [7], [6], [18], [28], [31]. For example, [3] (see Table 3 therein) as well as [28] (see Table 5.4 therein) reported computational results for dense QZOPs with up to 30 vertices; [23] (see Table 1 therein) reported computational results for dense QCPs with 40 vertices. [8] (see Table 1 therein) reported computational results for dense QZOPs with up to 50 vertices. For 75 vertices their algorithm only finds the exact optimum for 5 instances out of possible 10. For 100 vertices, they can only find the exact optimum for just one out of ten instances. Moreover, the general conclusion of all published computational studies can be summarized as follows [30]: "When the edge density is decreased, the polyhedral bound is slightly better. On the other hand, increasing the density makes the polyhedral bound poor." In other words, for all above mentioned methods, average calculation times grow as edge densities increase. Glover et al. [10] have reported computational experiments with the *adaptive memory tabu search* algorithm for the QZOP on dense graphs with 200 and 500 vertices, and they conclude that this problems are very difficult to solve by current *exact* methods: "Here, however, we have no proof of optimality, since these problems are beyond the scope of those that can be handled within practical time limits by exact algorithms" [10]. Using the DCA(PPA0) (see [14]) an exact global optimum of the QCP instances from [23] with 80 vertices on dense graphs, was found within 0.22 seconds on a PC with a 133 Mhz processor. In the next section we report computational results with the DCA(PPA3) for the QCP instances from [23] including instances up to 500 vertices on dense graphs.

6. Computational Experiments for the QCP with the DCA(PPAr)

In [14] we have restricted our computational experiments with the number of vertices up to 80 for the QCP instances, since instances of that size have been considered in [23]. For these instances, we have shown that the average calculation times grow exponentially when the number of vertices increases and reduce exponentially with increasing values of the density. For example, an exact global optimum of the QCP with 80 vertices and 100% density, was found within 0.22 sec on a personal computer with a 133 Mhz processor but for QCP instances with 80 vertices and 10% density 28.12 sec is required on the same computer. Therefore, the DCA(PPA0) was more efficient for QCP instances defined on dense graphs. However, we did not answer the following questions.

1. What are the largest QCP instances from [23] defined on dense graphs that can be solved by DCA(PPA0) within reasonable time?
2. Is it possible to increase the sizes of the QCP instances that can be solved by data correcting algorithms using any modification of the DCA(PPA0)?

We answer these questions in the remainder of this section. Since most of the published computational experiments with QZOP instances restrict the maximum CPU time to 10 minutes, we also use this as an upper limit for our computations.

For sake of completeness. we show below that the goal function of the QCP is submodular by using the following definition of the submodularity [26]

$$z(S+j) - z(S) \geq z(S+k+j) - z(S+k), \quad \text{for all } S \subseteq N.$$

Lemma 12. *The goal function $z(S)$ of the QCP is submodular.*

Proof. We know that $z(S) = \sum_{i\in S} p_i - \frac{1}{2}\sum_{i,j\in S} q_{ij}$. Now

$$\begin{aligned} & z(S+j) - z(S) - (z(S+k+j) - z(S+k)) \\ = & \sum_{i\in S+l} p_i - \frac{1}{2}\sum_{i,j\in S+l} q_{ij} - (\sum_{i\in S} p_i - \frac{1}{2}\sum_{i,j\in S} q_{ij}) \\ & -\Big(\sum_{i\in S+k+l} p_i - \frac{1}{2}\sum_{i,j\in S+k+l} q_{ij} - (\sum_{i\in S+k} p_i - \frac{1}{2}\sum_{i,j\in S+k} q_{ij})\Big) \\ = & -\sum_{i,j\in S+l} q_{ij} + \sum_{i,j\in S} q_{ij} + \sum_{i,j\in S+k+l} q_{ij} - \sum_{i,j\in S+k} q_{ij} \\ = & q_{kl} + q_{lk} \geq 0, \end{aligned}$$

since $q_{ij} \geq 0$ for all $i, j \in N$. This completes the proof ■

We have tested the DCA(PPAr) for QCPs on a Pentium processor running at 300 Mhz with 64 MB memory. All algorithms are implemented in Delphi 3.

The largest portion of the computation time is used to calculate the values of $d_k^+(S)$ and $d_k^-(T)$, since they are calculated rather frequently in the course of the algorithm. In case of the QCP we may calculate, for example, the value of $d_k^+(S)$, by calculating, at the first step, the expressions of $z(S+k) = \sum_{i\in S+k} p_i - \frac{1}{2}\sum_{i,j\in S+k} q_{ij}$ and $z(S) = \sum_{i\in S} p_i - \frac{1}{2}\sum_{i,j\in S} q_{ij}$, and, at the second step, $d_k^+(S) = z(S+k) - z(S)$.

However, we can simplify the calculating of $d_k^+(S)$ as follows:

$$\begin{aligned} d_k^+(S) &= z(S+k) - z(S) \\ &= \sum_{i\in S+k} p_i - \frac{1}{2}\sum_{i,j\in S+k} q_{ij} - (\sum_{i\in S} p_i - \frac{1}{2}\sum_{i,j\in S} q_{ij}) \\ &= p_k - \frac{1}{2}\sum_{i,j\in S} q_{ij} + \frac{1}{2}\sum_{i,j\in S} q_{ij} - \frac{1}{2}\sum_{i\in S} q_{ik} - \frac{1}{2}\sum_{j\in S} q_{kj} \\ &= p_k - \frac{1}{2}\sum_{i\in S} q_{ik} - \frac{1}{2}\sum_{j\in S} q_{kj}. \end{aligned}$$

Since $q_{kk} = 0$ and $q_{ij} = q_{ji}$ the last expression can be rewritten as

$$d_k^+(S) = p_k - \sum_{i\in S} q_{ik}.$$

Similarly, $d_k^-(T)$ can be written as

$$d_k^-(T) = \sum_{i\in T} q_{ik} - p_k.$$

Note that values of $d_k^+(S)$ and $d_k^-(T)$ must be calculated for successive sets such that $S, S+t_0, S+t_0+t_1$ etc., and $T, T-t_0, T-t_0-t_1$ etc. Hence, we can compute these values using incremental effort as follows

$$d_k^+(S+t) = d_k^+(S) - q_{tk} \text{ and } d_k^-(T-t) = d_k^-(T) - q_{tk}.$$

If we compare the two implementations of the DCA(PPAr), namely with the direct calculation of differences between $d_k^+(S+t)$and $d_k^-(T-t)$, and with the incremental method above, then computational experiments show that the average computational time is reduced, on average, by a factor of 2.

As problem instances we use randomly generated connected graphs having from 50 to 500 vertices and densities 10–100% which are 'statistically' similar to [23]. The *density* is defined as

$$d = \frac{|E|}{n(n-1)/2} \cdot 100\%,$$

where $|E|$ is the number of generated edges and $n(n-1)/2$ is the number of edges in a complete simple graph. The data p_i and q_{ij} are uniformly distributed with $p_i \in [0, 100]$ and $q_{ij} \in [1, 100]$. So, we may compare our computational results (see also [14]) to those obtained by [23].

First we look at calculation times of the DCA(PPAr) needed for problems varying from 50 to 100 vertices. Since the DCA(PPAr) easily finds an optimal solution to instances for which an optimal solution is located close to the top or to the bottom of the Hasse diagram, we use the *distance*

$$dist(|I|, n/2) = \frac{||I| - n/2|}{n/2} \cdot 100\%$$

between the calculated optimal solution I and the level $n/2$ of the "main diagonal" of the Hasse diagram in percentages as one of parameters of the "hardness" of our instances by solving them to optimality by the DCA(PPAr).

Intuitively, it is clear that the DCA(PPAr) applied to instances with low values of distance requires more calculation time than the DCA(PPAr) applied to instances with high values of distance. Empirically we have found (see [14]) that for sparse instances from [23] the distance from the "main diagonal" of the Hasse diagram is low (see Figure 1). For sparse instances with densities less than 20% DCA(PPA0) outperforms the branch-and-cut algorithm from [23], often with speeds that are ten times faster. For non-sparse instances with density more than 40% DCA(PPA0) is often 100 times faster than the branch-and-cut algorithm. Figure 1 shows that the distance grows when the density of instances increases. Therefore, we can expect a decrease in the average calculation time [14] when the density of instances increases (see Figure 2). Figure 2 shows that the natural logarithm of the average calculation time is approximately linear, i.e. the average calculation time grows exponentially with the size of instances. Moreover, this increase is faster for sparse instances than for dense ones.

We also study the effect of varying the number r of levels of the PPAr on the average calculation time of the DCA(PPAr). Figure 3 shows that searching one or more levels deep does not decrease the average calculation time for sparse instances (density< 1.0). The lowest average calculation time is achieved at level 3 for instances of complete graphs (density$= 1.0$). This fact is explained for all cases by the number of subproblems generated for different r values (see Figure 4). In Figure 4 it can be seen that in all cases the number of subproblems decreases when we search deeper, but the percentage of decrease in the number of subproblems is only 14% for levels 0 through 5 for instances with density of 70% while it is 91% for instances with density 100%. Therefore the profit accrued from decreasing the number of subproblems is spent on the additional cost computations for PPAr with higher r values. In particular, for dense graphs the balance is positive for search levels 3 and 4. This effect holds also for larger instances (see Figure 5).

In the second part of experiments we consider instances of the QCP with number of vertices ranging from 100 to 500 and densities varying between 10% and 100%. We try to solve these instances exactly, and with a prescribed accuracy of 5% within 10 minutes. Table 1 reports calculation times in seconds for exact and approximate solutions with DCA(PPA0) and Table 2 reports the same entries with DCA(PPA3). The entries marked with '*' could not be solved within 10 minutes. In all experiments of the second part, we note the exponentially increasing calculation times with increasing of sizes and decreasing of densities. Therefore QCP instances with 500 vertices and densities between 90% and 100% are the largest instances which can be solved by the DCA(PPA3) within 10 minutes on a standard personal computer. The impact of the diagonal dominance (see [14]) for instances of the QCP is similar to that in our previous experiments.

7. Concluding Remarks

In this paper we study the effect of increasing the depth of search in a Preliminary Preservation Algorithm on the performance of data correcting for the Quadratic Cost Partitioning problem, and by extension to the class of submodular functions. Theorems 9a and 9b, which can be considered as generalizations of Theorems 2a and 2b form the basis of the DCA(PPAr) algorithm. Theorem 9a states that if an interval $[S, T]$ is split into $[S, T - k]$ and $[S + k, T]$, then the maximum value of

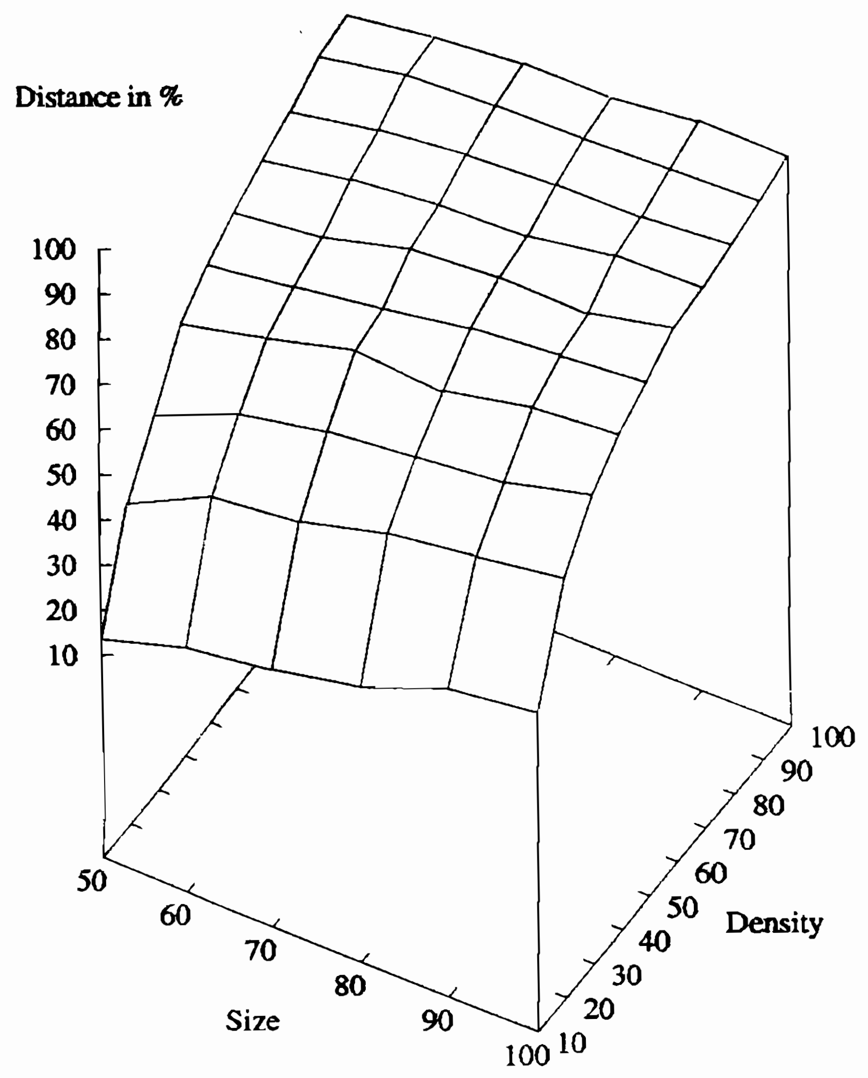


FIGURE 1. $dist(|I|, n/2)$ for instances of the QCP with 50–100 vertices and densities 10%–100%

TABLE 1. Average calculation times for DCA(PPA0) in seconds for QCP instances with 100-500 vertices and densities 10%-100% within 10 minutes

Problem	$n = 100$		$n = 200$		$n = 300$		$n = 400$		$n = 500$	
	$\varepsilon_0 = 0\%$	$\varepsilon_0 = 5\%$	$\varepsilon_0 = 0\%$	$\varepsilon_0 = 5\%$	$\varepsilon_0 = 0\%$	$\varepsilon_0 = 5\%$	$\varepsilon_0 = 0\%$	$\varepsilon_0 = 5\%$	$\varepsilon_0 = 0\%$	$\varepsilon_0 = 5\%$
100	0.831	0.752	64.372	38.351	340.681	229.396	1894.811	1162.396	*	*
90	1.027	0.916	78.614	49.926	794.037	583.754	3505.892	1996.544	*	*
80	1.784	1.108	217.898	162.455	2681.973	1875.603	*	3604.715	*	*
70	2.498	1.593	631.492	376.629	*	3165.384	*	*	*	*
60	3.509	2.874	1414.103	895.426	*	*	*	*	*	*
50	9.382	5.931	*	1937.673	*	*	*	*	*	*
40	17.245	10.327	*	*	*	*	*	*	*	*
30	48.013	22.209	*	*	*	*	*	*	*	*
20	195.82	74.841	*	*	*	*	*	*	*	*
10	446.293	95.122	*	*	*	*	*	*	*	*

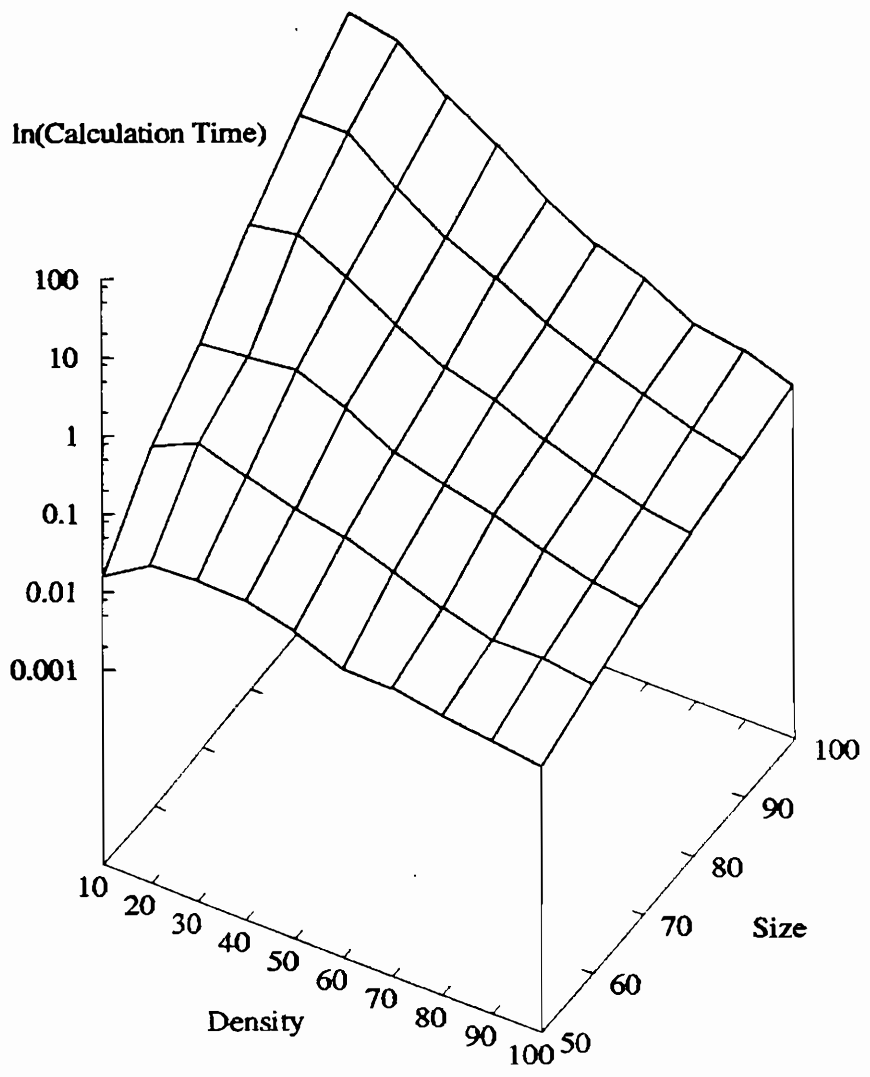


FIGURE 2. Natural logarithm of the average calculation time (in seconds) for instances of the QCP with 50–100 vertices and densities 10%–100%

TABLE 2. Average calculation times for DCA(PPA3) in seconds for QCP instances with 100-500 vertices and densities 10%-100% within 10 minutes

Problem	$n = 100$		$n = 200$		$n = 300$		$n = 400$		$n = 500$	
	$\varepsilon_0 = 0\%$	$\varepsilon_0 = 5\%$	$\varepsilon_0 = 0\%$	$\varepsilon_0 = 5\%$	$\varepsilon_0 = 0\%$	$\varepsilon_0 = 5\%$	$\varepsilon_0 = 0\%$	$\varepsilon_0 = 5\%$	$\varepsilon_0 = 0\%$	$\varepsilon_0 = 5\%$
100	0.098	0.094	2.63	2.444	18.316	17.179	85.827	85.096	229.408	222.883
90	0.138	0.118	3.824	3.607	37.931	34.972	173.063	166.996	624.925	608.755
80	0.28	0.228	9.506	8.186	98.69	89.685	679.914	580.789	*	*
70	0.393	0.304	17.643	15.693	413.585	364.48	*	*	*	*
60	0.731	0.517	86.33	72.931	*	*	*	*	*	*
50	1.752	1.298	345.723	267.445	*	*	*	*	*	*
40	3.457	2.179	*	*	*	*	*	*	*	*
30	11.032	5.88	*	*	*	*	*	*	*	*
20	47.162	17.477	*	*	*	*	*	*	*	*
10	70.081	12.196	*	*	*	*	*	*	*	*

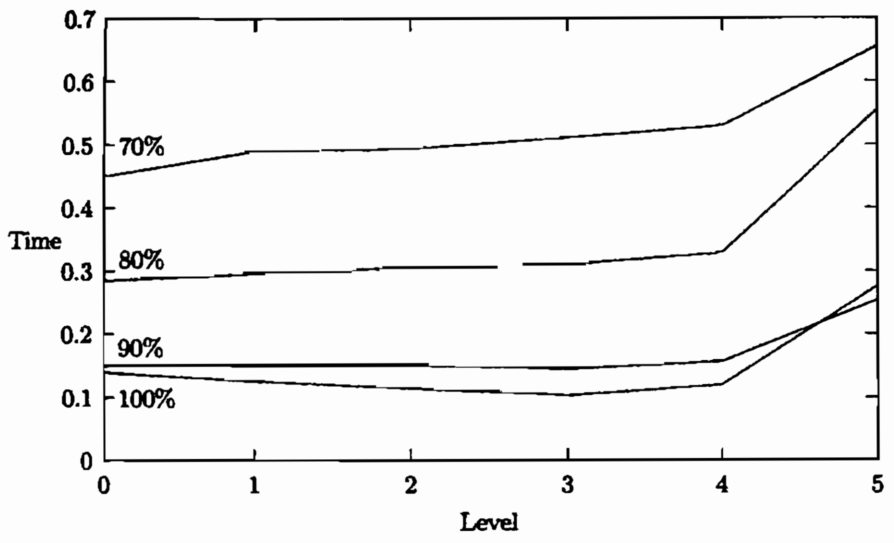


FIGURE 3. Average calculation times (in seconds) for values of r for instances of the QCP with 100 vertices and densities 70–100%

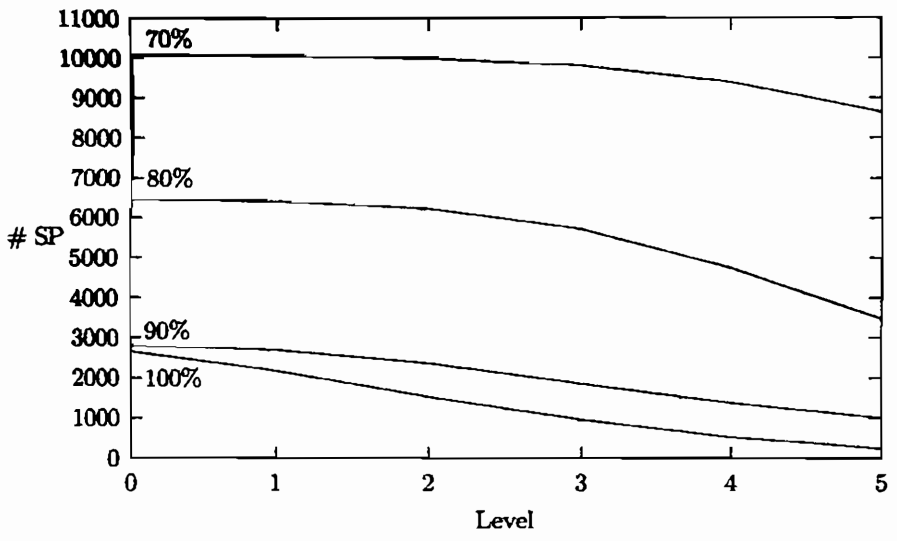


FIGURE 4. The number of generated subproblems against the level r for instances of the QCP with 100 vertices and densities 70%–100%

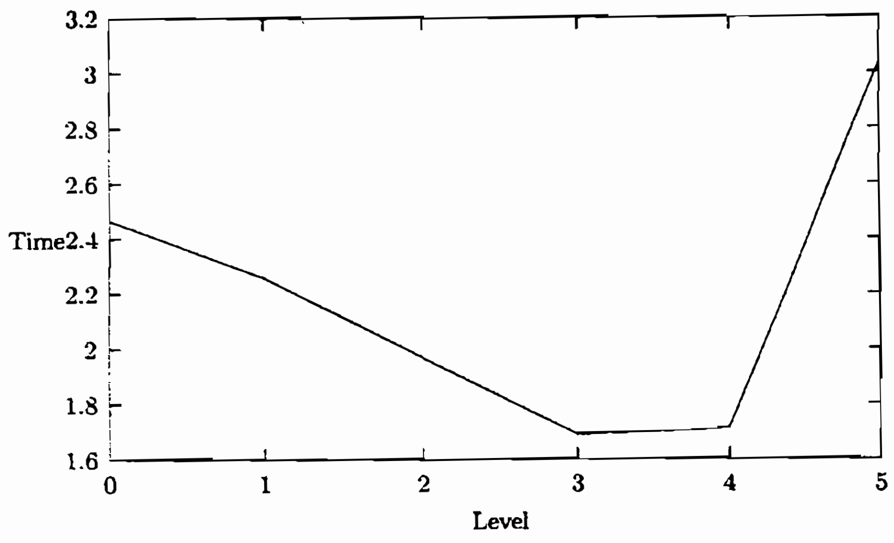


FIGURE 5. The average calculation time (in seconds) against the level r for instances of the QCP with 200 vertices and density 100%

all differences between the submodular function values z at levels $r+1$ and r is an *upper bound* to the difference between the unknown optimal value on the discarded subinterval and the maximum of the unknown value of the preserved subinterval and the maximum value of $z(I)$ on r levels of the Hasse diagram. Theorem 9b can be explained in a similar manner. These upper bounds are used to implicitly "correct" the value of z by correcting the value of the current accuracy.

We have tested the DCA(PPAr) on the QCP instances which are statistically similar to the instances in [23]. We show that the distance of an optimal solution to the main diagonal of the Hasse diagram is a good measure of the difficulty of a QCP instance at least for the DCA(PPAr). This distance increases at a rate slower than linear with increasing values of the density, for a fixed size of QCP instances. The instances with distances between 0% and 20% can be categorized as "hard", between 30% and 60% as "difficult", and between 70% and 100% as "easy". In all the instances tested, the average calculation time increases exponentially with decreasing density values for all prescribed accuracy values. This behavior differs from the results of the branch and cut algorithm in [23], in which calculation times increase when densities increase. This effect is also demonstrated for all algorithms based on linear programming (see, e.g., [3], [28], and [30]). This behavior makes the DCA an algorithm of choice for QCP instances with high densities. Our experiments with different values of r in the PPAr show that for the QCP instances from [23], the best r values are 3 and 4. This effect becomes more pronounced when the density of the corresponding instances approach 100%.

Recently, Glover et al. [10] reported their computational experiments for binary quadratic programs with *adaptive memory tabu search* procedures. They showed that their so called "c" problems with $n = 200$ and $n = 500$ (which are statistically equivalent to the [23] instances defined on dense graphs) to be the most challenging problems reported in the literature to date " ... far beyond the capabilities of current exact methods and challenging as well for heuristic approaches".

We have used the DCA(PPA3) to solve QCP instances with up to 500 vertices on dense graphs within 10 minutes using a personal computer with 64 MB RAM operating at 300 MHz. Note that the largest QCP instances solved by the DCA based on the two neighboring levels in the Hasse diagram within 10 min on the same computer had 300 vertices. Since the data-correcting approach is efficient for solving large QCP instances defined on the dense graphs, while branch-and-cut algorithms are efficient for solving large instances on sparse graphs, it will be interesting to investigate hybrids of the two for solving large instances of the QCP for all densities.

Acknowledgements. The authors would like to thank Marius de Vink and A. Naivelt for their help in the preparation of this paper.

REFERENCES

[1] J.E. Beasley. 1993. Lagrangean heuristics for location problems. European J. Opernl. Res. vol 65, pp 383–399.

[2] F. Barahona, M. Grötschel, M. Junger, and G. Reinelt. 1988. An application of combinatorial optimization to statistical physics and circuit layout design. Oper. Res. vol 36(3), pp 493–512.

[3] F. Barahona, M. Junger, and G. Reinelt. 1989. Experiments in quadratic 0-1 programming. Math. Prog. vol 44, pp 127–137.

[4] A. Billionet and A. Sutter. 1994. Minimization of quadratic pseudo-Boolean functions. European J. Opernl. Res. vol 78, pp 106–115.

[5] E. Boros and P. Hammer. 1993. Cut-polytopes, boolean quadric polytopes and nonnegative quadratic pseudo-boolean functions. Math. of Oper. Res. vol 18(1), pp 245–253.

[6] M.W. Carter. 1984. The indefinite zero-one quadratic problem. Discr. Appl. Math. vol 7, pp 23–44.

[7] R. Carragan and P. Pardalos. 1990. An exact algorithm for the maximum clique problem. Oper. Res. Lett. vol 9, pp 375–382.

[8] P. Chardaire and A. Sutter. 1995. A decomposition method for quadratic zero-one programming. Mgmt. Sc. vol 41(4), pp 704–712.

[9] V.P. Cherenin. 1962. Solving some combinatorial problems of optimal planning by the method of successive calculations. in Proceedings of the Conference on Experiences and Perspectives of the Application of Mathematical Methods and Electronic Computers in Planning, Mimeograph, Novosibirsk (in Russian).

[10] F. Glover, G. A. Kochenberger, and B. Alidaee. 1998. Adaptive memory tabu search for binary quadratic programs. Man. Sc. vol 44(3), pp 336–345.

[11] B. Goldengorin. 1983. A correcting algorithm for solving some discrete optimization problems. Soviet Math. Dokl. vol 27, pp 620–623.

[12] B. Goldengorin. 1995. Requirements of Standards: Optimization Models and Algorithms. Russian Operations Research, Hoogezand, The Netherlands.

[13] B. Goldengorin and G. Gutin. 1998. Polynomially solvable cases of the supermodular set function minimization problem. Research Report TR/6/98. Department of Mathematics and Statistics, Brunel University, Uxbridge, Middlesex, United Kingdom.

[14] B. Goldengorin, G. Sierksma, G.A. Tijssen, and M. Tso. 1999. The data-correcting algorithm for the minimization of supermodular functions. Man. Sc. vol 45(11), pp 1539–1551.

[15] R.P. Grimaldi. 1994. Discrete and Combinatorial Mathematics: An Applied Introduction. Third Edition, Addison-Wesley Publishing Company, New York.

[16] P. Hammer. 1965. Some network flow problems solved with pseudo-boolean programming. Oper. Res. vol 13, pp 388–399.

[17] J. Hastad 1997. Some optimal in-approximability results. Proceedings 29th Annual ACM Symposium on Theory of Computation, pp 1–10.

[18] F. Kalantari and A. Bagchi. 1988. An algorithm for quadratic zero one programming. Technical Report LCSR-TR-112, Dept. of Computer Science, Rutgers University.

[19] R.M. Karp. 1972. Reducibility among combinatorial problems. In: R.E. Miller, J.W. Thatcher (Eds.). Complexity of computer computations., New York: Plenum Press, pp 85–103.

[20] V.R. Khachaturov. 1968. Some problems of the successive calculation method and its applications to location problems. Dissertation. Central Economics & Mathematics Institute Russian Academy of Sciences, Moscow (in Russian).

[21] V.R. Khachaturov. 1989. Mathematical methods of regional programming. Moscow, Nauka, (in Russian).

[22] M. Laurent. 1997. Max-Cut Problem. In: M. Dell'Amico, F. Maffioli, and S. Martello (Eds.). Annotated Bibliographies in Combinatorial Optimization. John Wiley & Sons, Ltd. pp 241–259.

[23] H. Lee, G.L. Nemhauser, and Y. Wang. 1996. Maximizing a submodular function by integer programming: Polyhedral results for the quadratic case. European J. Opernl. Res. vol 94, pp 154–166.

[24] L. Lovasz. 1983. Submodular functions and convexity. In: A. Bachem, M. Grötschel, and B. Kort (Eds.). Mathematical Programming: The State of the Art., Springer-Verlag, Berlin, Germany, pp 235–257.

[25] M. Minoux. 1977. Minoux M. Accelerated greedy algorithms for maximizing submodular set functions. In: j. Stoer (Ed.). Actes Congres IFIP. Berlin: Springer Verlag, pp 234–243.

[26] G.L. Nemhauser, L.A. Wolsey, and M.L. Fisher. 1978. An analysis of approximations for maximizing submodular set functions-1. Math. Prog. vol 14, pp 265–294.

[27] M. Padberg. 1989. The boolean quadratic polytope: some characteristics, facets and relatives. Math. Prog. vol 45, pp 139–172.

[28] P.M. Pardalos and G.P. Rodgers. 1990. Computational aspects of a branch and bound algorithm for quadratic zero-one programming. Computing vol 40, pp 131–144.

[29] S. Poljak and F. Rendl. 1994. Node and edge relaxations of the max-cut problem. Computing vol 52, pp 123–137.

[30] S. Poljak and F. Rendl. 1995. Solving the max-cut problem using eigenvalues. Discr. Appl. Math. vol 62, pp 249–278.

[31] A.C. Williams. 1985. Quadratic 0–1 programming using the roof dual with computational results. RUTCOR Research Report 8-85. Rutgers University.

[32] D.P. Williamson. 1988. Lecture Notes on Approximation Algorithms. IBM Research Report, IBM Research Division, T.J. Watson Research Center, Yorktown Heights, New York.

Department of Econometrics and Operations Research, University of Groningen, The Netherlands
E-mail address: B.Goldengorin@eco.rug.nl

P&QM Area, Indian Institute of Management, Ahmedabad, India
E-mail address: diptesh@iimahd.ernet.in