Solving large instances of the quadratic cost of partition problem on dense graphs by data correcting algorithms
Goldengorin, Boris; Vink, Marius de
Solving Large Instances of the Quadratic Cost Partition Problem on Dense Graphs by Data-Correcting Algorithms: a Computational Study

Boris Goldengorin, Marius de Vink

20th August, 1999

Abstract

The Data-Correcting Algorithm (DCA) corrects the data of a hard problem instance in such a way that we obtain an instance of a well solvable special case. For a given prescribed accuracy of the solution, the DCA uses a branch and bound scheme to make sure that the solution of the corrected instance satisfies this given prescribed accuracy. We describe the “hardness” of randomly generated instances of the Quadratic Cost Partition Problem (QCP) by the density of the corresponding graphs as well as by the cardinality of an optimal solution of the QCP. We study the behaviour of the DCAs through the number of search levels of the so called Preliminary Preservation Algorithm and average computational times. We report average computational times of the DCA for instances of dense graphs with the number of vertices up to 500 which can be solved on a standard PC within 10 minutes.

Keywords: global maximum, submodular functions, quadratic programming

* Department of Econometrics and Operations Research, University of Groningen, P.O.Box 800, 9700 AV Groningen, The Netherlands. Corresponding author. Tel. +31-50-363 3794 fax: +31-50-363 3720, e-mail: B.Goldengorin@eco.rug.nl
1. Scope and purpose

Given a graph with nonnegative weights on the edges and real numbers as weights of vertices. For every set of vertices, the difference between the sum of vertex weights of this set and the one-half of the sum of all edges from the same set of vertices is the goal function which must be maximized in the Quadratic Cost Partition Problem (QCP). The QCP is NP-hard and applications of the QCP include capital budgeting, time tabling, communication scheduling, statistical physics, design of printed circuit boards and VLSI design. To our best knowledge, the known algorithms solve the QCP on dense graphs with the number of vertices being at most 50. The purpose of this paper is to present a new data-correcting algorithm which solves the QCP on dense graphs with number of vertices up to 500 within 10 min on a standard PC.

2. Introduction

The Quadratic Cost Partition Problem (QCP) can be described as follows. Given nonnegative real numbers $q_{ij}$ and real numbers $p_i$ with $i, j \in V = \{1, 2, \ldots, n\}$, the QCP is the problem of finding a subset $S \subset V$ such that the function

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

will be maximized. As a special case we have the Max-Cut Problem (MCP), to be described as follows. Consider an edge weighted undirected graph $G(V, E)$ with edge weights $w_{ij} \geq 0, i,j \in E$. Define a cut $\delta(T)$ as the edge set containing all the edges with one end in $T$ and the other end in $V \setminus T$. 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 $p_i = \sum_{i \in V} w_{ij}$ and $q_{ij} = 2w_{ij}$.

The QCP and the MCP arise in many real world applications (see [17] and references within) such as capital budgeting, time tabling, communication scheduling, statistical physics, design of printed circuit boards and VLSI design (see also Barahona et al. [1], Carter [3], and Lee et al. [18]). Since the MCP is a special case of the QCP, the QCP is also NP-hard, Karp [14]. The best known $\alpha$-approximation algorithm for MCP gives $\alpha = 0.87856$ (Williamson [25]). On the negative side though, Hastad
[13] 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 Hammer [12]) in terms of an unconstrained quadratic zero-one programming problem (QZOP) is the following pseudo-Boolean formulation:

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

Since $x_i^2 = x_i$, we can assume that the diagonal of $Q = ||q_{ij}||$ is zero. A mixed-integer programming (MIP) formulation can be found in Padberg [21]. In this formulation the quadratic term is replaced by a linear one and a number of linear constraints:

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

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

Another MIP formulation is given in Lee et al. [18]:

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

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

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

An advantage of the latter formulation over Padberg’s formulation [21] is the smaller number of variables, although an exponential number of constraints is required. The exponential number of constraints makes it impossible to solve the full formulation.
for large instances. In order to overcome this difficulty Lee et al. [18] start with only a small subset of the vertex set constraints \( \lambda \geq \sum_{i,j \in E(T) \cup E(T')} q_{ij} (x_i + x_j - 1) \) and generate violated ones as soon as 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. Boros and Hammer [2] have shown that the corresponding separation problems are polynomially solvable for a wide class of QZOPs.

The methods of computational studies of the QCP or the MCP can be classified into the following groups [17]: branch and bound methods [22], linear programming based methods (branch and cut algorithms [1], [18], eigenvalue based methods, and approaches via semidefinite programming [23], [24]. We will not discuss all of these approaches but restrict 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. For example, Barahona et al. (see Table 3 in [1]) as well as Pardalos and Rodgers (see Table 5.4 in [22]) reported computational results for dense QZOPs with up to 30 vertices; Lee et al. (see Table 1 in [18]) reported computational results for dense QCPs with 40 vertices. Chardaire and Sutter (see Table 1 in [4]) 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 [4]. Moreover, the general conclusion of all published computational studies can be summarized as follows [24]: “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. Recently Glover et al.[6] 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” [6].

The main purpose of this paper is to provide a computational study of the behaviour of the Data-Correcting Algorithm (DCA) on large instances of the QCP for dense graphs with the surprising result that we have found an NP-hard class of the dense QZOPs, namely the dense QCPs, for which the DCA finds an exact optimal solution on a standard PC within 10 minutes.
We have organized our paper as follows. In Section 3 we show that the goal function of the QCP will be submodular if the weights of all edges of the corresponding graph are nonnegative (see Lemma 3.1) and weights of vertices are real numbers. We briefly review the so-called Preliminary Preservation Algorithm (PPA) (see Theorem 3.1 and Corollaries 3.1 and 3.2), which we use for determining the relevant polynomially solvable class of submodular functions (PP-functions). In Section 4 we present a generalization of the PPA, namely the PPA of order $r$ (PPAr) (see Theorems 4.1, 4.2, 4.3 and Corollaries 4.1, 4.2). The DCA is described in Section 5 and the results of the computational testings are analyzed in Section 6. Section 7 gives a number of concluding remarks.

3. The polynomially solvable case of the maximization of a submodular function

In this section we give a brief introduction to the polynomially solvable case for the maximization of submodular functions. We use the Preliminary Preservation Algorithm (PPA) [5] for the construction of the polynomially solvable case of submodular functions, called PP-functions. Details about the PPA and PP-functions can be found in Goldengorin et al. [10] (see also [8–9]).

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

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

Note that $[\emptyset, V] = 2^V$. Any interval $[S, T]$ is, in fact, a subinterval of $[\emptyset, V]$ if $\emptyset \subseteq S \subseteq T \subseteq V$; notation $[S, T] \subseteq [\emptyset, V]$. In this paper we mean by an interval always a subinterval of $[\emptyset, V]$. Throughout this paper, it is assumed that $z$ attains a finite maximum value on $[\emptyset, V]$. The function $z$ is called submodular on $[S, T]$ if for each $I, J \in [S, T]$ it holds that

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

The following definition of the submodularity

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

is equivalent to the previous one [20].
**Lemma 3.1**  
*The goal function* $z(S)$ *of the QCP is submodular.*

**Proof.** Let us substitute the expression of $z(S) = \sum_{i \in S} p_i - \frac{1}{2} \sum_{i,j \in S} q_{ij}$ to the last definition of the submodularity.

\[
\sum_{i \in S} p_i - \frac{1}{2} \sum_{i,j \in S} q_{ij} - \left( \sum_{i \in S} p_i - \frac{1}{2} \sum_{i,j \in S} q_{ij} \right) \geq
\]

\[
\sum_{i \in S+k} p_i - \frac{1}{2} \sum_{i,j \in S+k} q_{ij} - \left( \sum_{i \in S+k} p_i - \frac{1}{2} \sum_{i,j \in S+k} q_{ij} \right).
\]

Cancelling out terms involving $p_i$ gives

\[
- \sum_{i,j \in S} q_{ij} + \sum_{i,j \in S+k} q_{ij} \geq - \sum_{i,j \in S+k} q_{ij} + \sum_{i,j \in S+k} q_{ij}.
\]

After all bookkeeping we have

\[
q_{kl} + q_{lk} \geq 0.
\]

Since $q_{ij} \geq 0$ for all $i, j \in V$, the proof is completed. \qed

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 maximum value of the submodular function $z$ on the interval $[S, T] \subseteq [\emptyset, V]$ is denoted by $z^*[S, T]$. The following theorem 1 and corollary 1 [10] are the base of the PPA.

**Theorem 3.1**  
Let $z$ be a submodular function on $[S, T] \subseteq [\emptyset, V]$ and let $k \in T \setminus S$. Then the following assertions hold.

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

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

**Corollary 3.1**  
*(preservation rules of order zero).* Let $z$ be a submodular function on $[S, T] \subseteq [\emptyset, V]$ and let $k \in T \setminus S$. Then the following assertions hold.
First Preservation Rule (FPR)
(a) If \( z(S + k) - z(S) = d^+_k(S) \leq 0 \), then \( z^*[S, T] = z^*[S, T - k] \geq z^*[S + k, T] \).

Second Preservation Rule (SPR)
(b) If \( z(T - k) - z(T) = d^-_k(T) \leq 0 \), then \( z^*[S, T] = z^*[S + k, T] \geq z^*[S, T - k] \).

Corollary 3.2
Let \( z \) be a submodular function on \( [S, T] \subseteq [\emptyset, V] \) and let \( k_i \in T \setminus S, i \in R = \{1, \ldots, r\} \). Then the following assertions hold.

First Extended Preservation Rule
(a) If \( d^+_k(S) \leq 0 \) for all \( i \in R \), then \( z^*[S, T] = z^*[S, T \setminus \bigcup_{i \in R} k_i] \).

Second Extended Preservation Rule
(b) If \( d^-_k(T) \leq 0 \) for all \( i \in R \), then \( z^*[S, T] = z^*[S \cup \bigcup_{i \in R} k_i, T] \).

Proof.
We prove only part (a) since the proof of (b) is similar. Applying the FPR does not change the values \( d^+_k(S) \) since only the set \( T \) changes and not the set \( S \). Let us apply the FPR with \( d^+_k(S) \leq 0 \), then the interval \( [S, T] \) will be reduced to the interval \( [S, T - k_1] \). The values \( d^+_k(S) \), \( i \in R - k_1 \) have not changed for the new interval \( [S, T - k_1] \) and hence after applying the FPR with \( d^+_k(S) \leq 0 \), 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 3.2 it is often possible to exclude a large part of \( [\emptyset, V] \) from consideration when determining a global maximum of \( z \) on \( [\emptyset, V] \). The PPA determines a subinterval \( [S, T] \) of \( [\emptyset, V] \) that certainly contains a global maximum of \( z \), whereas \( [S, T] \) cannot be made smaller by using the preservation rules of Corollary 3.2. In case the PPA stops with \( S = T \) then an optimal solution has been found, i.e. \( S \in \arg z^*[\emptyset, V] \).
The Preliminary Preservation Algorithm

**Procedure** PPA($X, Y; S, T$)

**Input:** A submodular function $z$ on interval $[X, Y]$ of $[\emptyset, V]$  
**Output:** The subinterval $[S, T] \subseteq [X, Y]$ such that $z^*[S, T] = z^*[X, Y]$ and \( \min\{d^+_i(S), d^-_i(T) \mid i \in T \setminus S\} > 0 \).

**Step 0:** $S \leftarrow X$, $T \leftarrow Y$, $S_1 \leftarrow \emptyset$, $T_1 \leftarrow \emptyset$.

**Step 1:** $S_1 \leftarrow \{ k \in T \setminus S \mid d^-_k(T) \leq 0 \}$, $S \leftarrow S \cup S_1$.

**Step 2:** $T_1 \leftarrow \{ k \in T \setminus S \mid d^+_k(S) \leq 0 \}$, $T \leftarrow T \setminus T_1$.

**Step 3:** If $S = T$, then end.

**Step 4:** If $T_1 \neq \emptyset$, then goto Step 1.

**Step 5:** If $S_1 \neq \emptyset$, then goto Step 2, else end.

Every time the interval $[S, T]$ is updated the conditions of Corollary 3.2 are met. Each new interval contains a global maximum because at each step $z^*[S, T] = z^*[X, Y]$. This shows the correctness of the PPA.

The following theorem can also be found in Goldengorin [8]. It provides an upper bound for the worst case complexity of the PPA; the complexity function is dependent only on the number of comparisons of pairs of values of $z$.

**Theorem 3.2** *The time complexity of the PPA is at most $O(n^2)$."

Any submodular function $z$ on $[X, Y]$ for which the PPA returns a global maximum for $z$ is called a PP-function. Theorem 3.3 gives 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., Grimaldi [11]) as the ground graph $G = (V, E)$ in which $V = [\emptyset, N]$ and a pair $(I, J)$ is an edge iff 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 $z(I)$; notation $G = (V, E, z)$. A local maximum $L \in [\emptyset, N]$ ($\overline{L} \in [\emptyset, N]$) is called a lower (respectively, upper) maximum if there is no another local maximum $L$ such that $L \subset L$ (respectively, $\overline{L} \subset L$).
If an interval \([L, \overline{L}]\) with \(L \subseteq \overline{L}\) has as its end points lower and upper maxima then the submodular function is constant on 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 \(V_0\). This subgraph consists of at least one connected component. We denote the connected components by \(H^j_0 = (V^j_0, E^j_0, z)\), 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 \(H^j_0\) is called a strict local maximum component (STC) if for each \(I = \{\overline{I}\} \in V^j_0\), for which there is an edge \((I, L)\) with \(L \in V^j_0\), we have \(z(I) < z(L)\).

A component \(H^j_0\) is called a saddle local maximum component (SDC) if for some \(I = \{\overline{I}\} \in V^j_0\), there exists an edge \((I, L)\) with \(L \in V^j_0\) such that \(z(I) = z(L)\).

Khachaturov [15] has observed that any global maximum belongs to an STC. The following Theorem 3.3 can be found in [8,10].

**Theorem 3.3**  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 \(I \in \emptyset \cup \{1, 2, 3\}\) 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 \([[1], \{1\}] \cup \{1, 3\} \cup \{2\} \cup \{2, 3\} \cup \{3\} \cup \{1, 3\} \cup \{2, 3\}\). 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 \(d^k_S(S) = p_k - \sum_{i \in S} q_{ik} > 0\) and \(d^k_T(T) = \sum_{i \in T} q_{ik} - p_k > 0\). If there is a sequence such that \(\emptyset \subseteq S_1 \subseteq S_2 \subseteq \ldots \subseteq S_p = T_{n-p} \subseteq T_{n-p+1} \subseteq \ldots \subseteq T_{n-1} \subseteq N\) with \(d^k_i(S_i) \leq 0\) and \(d^k_i(T_{n-i}) \leq 0\) for all \(i = 1, \ldots, p\) and some \(p \leq |T\setminus S|\) the PPA solves the QCP. Therefore the corresponding class of submodular functions can be expounded as PP-functions.

In the next section we determine a generalization of the PPA, called the **PPA of order r** (PPAr).
4. PPA of order $r$ [9]

The preservations rules in the PPA “look” only one level deep in the Hasse diagram. The following statements allow us to explore the solution space more than one level deep. This may be useful because of the additional possibilities of narrowing the original interval are obtained.

**Theorem 4.1** Let $z$ be a submodular function on $[S, T] \subseteq [\emptyset, V]$ and let $k \in T \setminus S$. Then the following assertions hold.

(a) For any $t^+_0(k) \in \arg \max \{d^+_{t_0} (S + t) : t \in T \setminus (S + k)\}$,
$$z^*[S + k, T] - \max \{z^*[S, T - k], z(S + k)\} \leq \max \{d^+_{t_0} (S + t^+_0(k)), 0\}.$$ 

(b) For any $t^-_0(k) \in \arg \max \{d^-_{t_0} (T - t) : t \in T \setminus (S + k)\}$,
$$z^*[S, T - k] - \max \{z^*[S + k, T], z(T - k)\} \leq \max \{d^-_{t_0} (T - t^-_0(k)), 0\}.$$ 

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

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:** $z(S + k) \leq z^*[S + k + t^+_1(k), T]$. Then $z^*[S + k, T] = z^*[S + k + t^+_1(k), T]$.

For any $k \in T \setminus [S + t^+_1(k)]$ Theorem 3.1(a) on the interval $[S + t^+_1(k), T]$ states that:
$$z^*[S + t^+_1(k) + k, T] - z^*[S + t^+_1(k), T - k] \leq d^+_{t^+_0} (S + t^+_1(k)),$$

i.e., in Case 1 this inequality can be written as follows:
$$z^*[S + k, T] - z^*[S + t^+_1(k), T - k] \leq d^+_{t^+_0} (S + t^+_1(k)),$$

or
Adding two maximum operations leads to the following inequality
\[ z^*[S + k, T] - \max[z^*[S, T - k], z(S + k)] \leq \max[d^+_k(S + t^+_1(k)), 0]. \]

Finally, \( d^+_k(S + t^+_1(k)) \leq d^+_k(S + t^+_0(k)) \) since \( d^+_k(S + t) \) was maximal for \( t^+_0(k) \).

Case 2: \( z(S + k) > z^*[S + k + t^+_1(k), T] \).

Then \( z^*[S + k, T] = z(S + k) \). Consider the inequality
\[ z(S + k) - \max[z^*[S, T - k], z(S + k)] \leq 0, \]

i.e.,
\[ z^*[S + k, T] - \max[z^*[S, T - k], z(S + k)] \leq 0. \]

Adding a maximum operation with \( d^+_k(S + t^+_0(k)) \) gives the required result
\[ z^*[S + k, T] - \max[z^*[S, T - k], z(S + k)] \leq \max[d^+_k(S + t^+_0(k)), 0]. \]

\[ \square \]

**Corollary 4.1** (preservation rules of order one). Let \( z \) be a submodular function on \( [S, T] \subseteq [\emptyset, V] \) and let \( k \in T \setminus S \). Then the following assertions hold.

**First Preservation Rule of Order One**
(a) If \( \max[d^+_k(S + t) : t \in T \setminus (S + k)] \leq 0 \), 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 \( \max[d^-_k(T) : t \in T \setminus (S + k)] \leq 0 \), then
\[ z^*[S, T] = \max[z^*[S + k, T], z(T - k)] \geq z^*[S, T - k] \]

In the following theorem we show 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 (see Corollary 3.1). 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. In this sense we will say that preservation rules of order one are not weaker than preservation rules of order zero.
Theorem 4.2   Preservations rules of order one are not weaker than preservations rules of order zero.

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

Assume that the preservation rule of order one is not applicable, i.e., \( \max \{ d^+_k (S + t) : t \in T \setminus (S + k) \} = d^+_k (S + t_0) > 0 \). 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 \( t \in T \setminus (S + k) \), i.e., \( \max \{ d^-_k (S + t) : t \in T \setminus (S + k) \} \leq 0 \).

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 because together with the preserved interval \([S + k, T]\) or \([S, T - k]\) we should preserve exactly one additional value \( z(T - k) \) or \( z(S + k) \), respectively. This property is also valid for preservation rules of order \( r \geq 1 \).

Instead of one level deep (order one) we may ‘look’ \( r \) levels deep (order \( r \)) 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

\[
M^+_r [S, T] = \{ I \in [S, T] : |I \setminus S| \leq r \},
\]

\[
M^-_r [S, T] = \{ I \in [S, T] : |T \setminus I| \leq r \}.
\]

The sets \( M^+_r [S, T] \) and \( M^-_r [S, T] \) are the collection of subsets that contain in the vicinity on one side of the sets \( S \) (the bottom of the corresponding Hasse subdiagram) and \( T \) (the top of the corresponding Hasse subdiagram) for \( r \) levels deep. Define further the collections of sets

\[
N^+_r [S, T] = M^+_r [S, T] \setminus M^+_r -1 [S, T],
\]

\[
N^-_r [S, T] = M^-_r [S, T] \setminus M^-_r -1 [S, T].
\]

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

\[
u^+_r [S, T] = \max \{ z(I) : I \in M^+_r [S, T] \}, \quad v^-_r [S, T] = \max \{ z(I) : I \in M^-_r [S, T] \}, \quad w^+_k [S, T] = \max \{ d^+_k (I) : I \in N^+_r [S + k, T] \} \text{ and } w^-_k [S, T] = \max \{ d^-_k (I) : I \in N^-_r [S, T - k] \}.
\]

Theorem 4.3   Let \( z \) be a submodular function on \([S, T] \subseteq [\emptyset, V]\) with \( k \in T \setminus S \) and let \( r \) is a positive integer. Then the following assertions hold.
(a) If $|N_j^+[S + k, T]| > 0$, then
$$
z^*[S + k, T] - \max\{z^*[S, T - k], v^+_j[S, T]\} \leq \max\{w^+_{rk}[S, T], 0\}.
$$

(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:

$$
[S, T] = M^+_j[S, T] \cup \bigcup_{I \in N_j^+[S, T]} [I, T].
$$

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

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

Let $I(k) \in \arg\max\{z^*[I + k, T] : I \in N_j^+[S, T]\}$, and let us consider two cases of the last equality:

Case 1. $z^*[I(k) + k, T] \geq v^+_j[S + k, T]$, and

Case 2. $z^*[I(k) + k, T] < v^+_j[S + k, T]$.

In the first case $z^*[S + k, T] = z^*[I(k) + k, T]$. For $I(k) \in N_j^+[S, T]$ Theorem 3.1 (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

$$
z^*[S + k, T] - \max\{z^*[S, T - k], v^+_j[S + k, T]\} \leq \max\{d^+_k(I(k)), 0\}.
$$
Since \( w_{kT}^+ [S, T] \) is the maximum of \( d_{kT}^+ (I) \) for \( I \in N_{kT}^+[S+k, T] \), 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 \( w_{kT}^+ [S, T] \) completes the proof of case (a)

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

\[\Box\]

**Corollary 4.2** (preservation rules of order \( r \)). Let \( z \) be a submodular function on \([S, T] \subseteq [\emptyset, V] \) and let \( k \in T \setminus S \). Then the following assertions hold.

**First Preservation Rule of Order \( r \)**

\( a) \quad \text{If } w_{kT}^+ [S, T] \leq 0, \text{ 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) \quad \text{If } w_{kT}^+ [S, T] \leq 0, \text{ 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 4.2 can be proved for preservation rules of order \( r - 1 \) and \( r \) as follows. Preservation rules of order \( r \) are not weaker than preserving rules of order \( r - 1 \).

Now we can describe the PPA of order \( r \) (PPAr). The PPA 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 PPA searches \( r \) levels deep. The PPA 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 Nemhauser et al. [20]), \( d_k^+(S) \geq d_k^+(S + t) \), if we choose \( d_k^+(S) \) as small as possible, then \( d_k^+(S + t) \) will be not 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 6 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 \( r = |Y \setminus X| \), 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 PParmin 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 \( t \in T \setminus (S + k) \) calculate \( d_k^+(S + t) \). If \( d_k^+(S + t) \leq 0 \) or \( r = 1 \) then \( \max d \leftarrow \max \{ \max d, d_k^+(S + t) \} \)
else call **PParplus** \( (S + t, T, k, r - 1, maxd) \)
*end*

**Procedure PParmin** \( (S, T, k, r, maxd) \)

*begin*
Calculate \( z(T - k) \). If \( z(B) < z(T - k) \) then \( B \leftarrow T - k \); For all \( t \in T \setminus (S + k) \) calculate \( d_k^-(T - t) \). If \( d_k^-(T - t) \leq 0 \) or \( r = 1 \) then \( \max d \leftarrow \max \{ \max d, d_k^-(T - t) \} \)
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 \([\emptyset, V]\)

**Output:** The subinterval \([S, T]\) and the set \( B \) such that

\[
z^\ast[X, Y] = \max\{z^\ast[S, T], z(B)\}
\]

\[
\min\{w^+_r[S, T], w^-_r[S, T]\} > 0 \text{ for all } k \in T \setminus S
\]

**Step 0:**

\[S \leftarrow X, T \leftarrow Y, S1 \leftarrow \emptyset, T1 \leftarrow \emptyset, B \leftarrow \emptyset.\]

**Step 1:** call \( \text{PPA}(X, Y; S, T); \) goto Step 2;

**Step 2:**

\[d^+ \leftarrow \max\{d_+^k(S) : k \in T \setminus S\}, d^- \leftarrow \max\{d_{-}^k(T) : k \in T \setminus S\};\]

\[\text{If } d^+ < d^- \text{ then goto Step 3 else goto Step 4.}\]

**Step 3:**

\[k \leftarrow \arg \max\{d_+^t(S) : t \in T \setminus S\};\]

\[\text{call PPA} \text{Arplus}(S, T, k, r, \text{max } d);\]

\[\text{If max } d \leq 0 \text{ then } T \leftarrow T - k, \text{ goto Step 1 else end.}\]

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.

5. The Data-Correcting Algorithm (DCA)

In this section we briefly describe the main idea and the structure of the DCA based on the PPA. We will call this DCA the DCA with PPAr and abbreviate to DCA(PPAr).

The description of the DCA(PPA) can be found in Goldengorin et al. [10]. 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 \([\emptyset, V]\) with \( S \neq T \) containing a maximum of \( z \) without knowing its exact location in \([S, T]\). In this case, the postcondition \( \min\{d^+_i(S), d^-_i(T) \mid i \in T \setminus S\} = \delta > 0 \) of the PPA is satisfied. The basic idea of the DCA is that if a situation occurs for which this postcondition 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 \( d^+_p(S) = \delta > 0 \) or \( d^-_p(T) = \delta > 0 \) for some \( k, p \in T \setminus S \). 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 we apply the correcting rules again and so on until the PPA finds an optimal solution. For \( k \in T \setminus S \) theorem 1 gives upper bounds for the values \( z^\ast[S, T] - z^\ast[S, T - k] \), namely, \( d^+_k(S) \), and for \( z^\ast[S, T] - z^\ast[S + k, T] \), 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 $J \in [\emptyset, V]$ such that $z^*[\emptyset, V] \leq z(J) + \varepsilon$, where $\varepsilon = \sum_{i=1}^n \delta_i$ with $\delta_i$ equal to either $d^+_i(S)$ or $d^-_i(T)$. These arguments lead to the following two upper bounds (see Khachaturov [15], Minoux [19]).

**Theorem 5.1** If $\min\{d^+_i(S), d^-_i(T) \mid i \in T\setminus S\} > 0$, then

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

and

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

Before the PPA stops there are a few options. First, if we would like to allow a certain prescribed accuracy, say $\varepsilon_0$, 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 $d^+_k(S) \geq d^+_k(S + t)$ of a submodular function is the base of the following branching rule (see [10]).

**Branching Rule:**

For $k \in \arg\max\{\max\{d^+_i(S), d^-_i(T)\} : i \in T\setminus S\}$, split the interval $[S, T]$ into two subintervals $[S + k, T]$, $[S, T - k]$, and use the prescribed accuracy $\varepsilon$ 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 than the optimal value found so far.

Due to Khachaturov [16], the upper bounds $ub_1$ and $ub_2$ from Theorem 5.1 can be tightened. Define the following sets of positive numbers: $d^+(S, T) = \{d^+_i(S) : d^+_i(S) > 0, i \in T \setminus S\}$ and $d^-(S, T) = \{d^-_i(T) : d^-_i(T) > 0, i \in T \setminus S\}$. Define further 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 $i = 1, \ldots, |T \setminus S|$. So, $d^+[1] \geq \ldots \geq d^+|[T \setminus S]|$ and $d^-[1] \geq \ldots \geq d^-|[T \setminus S]|$. Let $z^*[S, T, i] = \max[z(I) : N^+[S, T]]$ 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 $z^*[S, T] = \max[z^*[S, T, i] : i = 1, \ldots, |T \setminus S|]$ we have the following upper bound

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

Now we will describe the DCA. The DCA starts with a submodular function $z$ on the interval $[0, V]$ and the prescribed accuracy $\varepsilon_0$. 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 will be 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 postcondition 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 $\varepsilon$ with $d^+_k(S)$. In our example, if the value of the current accuracy of the interval $[S, T]$ is equal to $\varepsilon$, then after discarding the subinterval $[S + k, T]$ its value will be equal to $\varepsilon + d^+_k(S)$. These arguments show that the DCA didn’t change our submodular function explicit. 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 $z(I) + [z(J) + d^+_k(S)] \geq \max[z^*[S, T, i] : i = 1, \ldots, |T \setminus S|]$. After the DCA may discard the subinterval $[S + k, T]$, then the PPA stops with an interval $[S, T - k]$ and consequently, the upper bound $ub_1$ and $ub_2$ from Theorem 5.1 can be tightened.

18
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 more possibilities: either narrowing the current interval or to decreasing the current value of \( \varepsilon \).

6. **The QCP: Computational Experiments**

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 parts of calculation times are taken by the calculation of 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:

\[
d_k^+(S) = z(S + k) - z(S) =
\]

\[
\sum_{i \in S + k} p_i - \frac{1}{2} \sum_{i,j \in S + k} q_{ij} - (\sum_{i \in S} p_i - \frac{1}{2} \sum_{i,j \in S} q_{ij}) =
\]

\[
p_k - \frac{1}{2} \sum_{i,j \in S} q_{ij} + \frac{1}{2} \sum_{i,j \in S} q_{ij} - \frac{1}{2} \sum_{i \in S} q_{ik} - \frac{1}{2} \sum_{j \in S} q_{kj} =
\]

\[
p_k - \frac{1}{2} \sum_{i \in S} q_{ik} - \frac{1}{2} \sum_{j \in S} q_{kj}.
\]

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

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 use the previous value for calculating the next one 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 \( d_k^+(S) \) and \( d_k^-(T) \), and with the preliminary simplified expression of \( d_k^+(S) \), then computational experiments show that the average computational time is reduced, on average, by a factor 2.

As problem instances we use randomly generated connected graphs having from 50 to 500 vertices and densities 10–100%. 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 Goldengorin et al. [10]) to those obtained by Lee et al. [18].

First we look at calculation times of the DCA(PPAr) needed for problems varying from 50 to 500 vertices. 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. Intuitively, it is clear that the DCA applied to instances with distances close to 0% requires more calculation time than the DCA applied to instances with distances close to 100%. Figure 6.1 shows that the distance grows when the density of instances increases. Therefore, we can expect a decrease in the average calculation.
time [10] when the density of instances increases (see Figure 6.2). Figure 6.2 shows that the natural logarithm of the average calculation time is approximately linear. Hence, it is plausible that the average calculation time grows exponentially when the size of instances increases. Moreover, this increasing is more rapid for sparse instances than for dense ones. We also study the impact of the number $r$ of levels of the PPAr on the average calculation time of the DCA(PPAr). Figure 6.3 shows that searching one or more levels deep does not decrease the average calculation time for non-dense instances ($d < 1.0$). The smallest average calculation time is achieved at level 3 for instances of complete graphs ($d = 1.0$). This fact is explained for all cases by the number of generated subproblems for different levels $r$ (see Figure 6.4). In Figure 6.4 it can be seen that in all cases the number of subproblems is decreased when we search deeper, but the decrease percentage of the number of subproblems for levels 0 through 5 is only 14% for instances with density of 70% while it is 91% for instances with density 100%. Therefore the profit of decreasing the number of subproblems is spent on the additional costs of the average calculation time for the PPAr. More exactly, for dense graphs the balance is positive for search levels 3 and 4. This effect holds also for larger instances (see Figure 6.5).

In the second part of experiments we consider instances of the QCP with sizes between 100 and 500 vertices and densities between 10% and 100% which can be solved with a prescribed accuracy of up to 5% within approximately 10 minutes. Table 6.1 gives calculation times in seconds for an exact/approximate solutions (0%/5%). The entries in this table with * could not be solved within 10 minutes. All instances with sizes above 300 and densities below 50% could not be solved within 10 minutes and are not shown in Table 6.1. In all experiments of the second part, the effect of exponentially increasing calculation times with increasing of sizes and decreasing of densities is preserved. Therefore instances of the QCP with 500 vertices and densities between 90% and 100% are the largest instances which can be solved by the DCA(PPAr) within 10 minutes on a standard PC.

The impact of the “diagonal dominance” notion for instances of the QCP is the same as in our previous experiments [10].
Figure 6.1: $\text{dist}(|I|, n/2)$ for instances of the $QCP$ with 50–100 vertices and densities 10%–100%
Figure 6.2: Natural logarithm of the average calculation time (in seconds) for instances of the QCP with 50–100 vertices and densities 10%–100%
Figure 6.3: Average calculation times (in seconds) for values of $r$ for instances with size 100 and densities 70–100%

Figure 6.4: The number of generated subproblems against the level $r$ for instances of the QCP with 100 vertices and densities 70%–100%
Figure 6.5: The average calculation time (in seconds) against the level $r$ for instances of the $QCP$ with 200 vertices and density 100%
<table>
<thead>
<tr>
<th>Prob.</th>
<th>Dens</th>
<th>Time, average, secs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>100</td>
<td>0.098/0.094</td>
</tr>
<tr>
<td></td>
<td>90</td>
<td>0.138/0.118</td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>0.280/0.228</td>
</tr>
<tr>
<td></td>
<td>70</td>
<td>0.393/0.304</td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>0.731/0.517</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>1.752/1.298</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>3.457/2.179</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>11.032/5.880</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>47.162/17.477</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>70.081/12.196</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.630/2.444</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3.824/3.607</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9.506/8.186</td>
</tr>
<tr>
<td></td>
<td></td>
<td>17.643/15.693</td>
</tr>
<tr>
<td></td>
<td></td>
<td>86.330/72.931</td>
</tr>
<tr>
<td></td>
<td></td>
<td>345.723/267.445</td>
</tr>
<tr>
<td></td>
<td></td>
<td><em>/</em></td>
</tr>
<tr>
<td></td>
<td></td>
<td><em>/</em></td>
</tr>
<tr>
<td></td>
<td></td>
<td><em>/</em></td>
</tr>
</tbody>
</table>

Table 6.1: Average calculation times (in seconds) against prescribed accuracies of 0% and 5% for instances of the QCP with 100–500 vertices and densities 10%–100% within 10 min.

<table>
<thead>
<tr>
<th>Prob.</th>
<th>Dens</th>
<th>Time, average, secs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>300</td>
<td>18.316/17.179</td>
</tr>
<tr>
<td></td>
<td>90</td>
<td>37.931/34.972</td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>98.690/89.685</td>
</tr>
<tr>
<td></td>
<td>70</td>
<td>413.585/364.480</td>
</tr>
<tr>
<td></td>
<td>60</td>
<td><em>/</em></td>
</tr>
<tr>
<td></td>
<td></td>
<td>85.827/85.096</td>
</tr>
<tr>
<td></td>
<td></td>
<td>173.063/166.996</td>
</tr>
<tr>
<td></td>
<td></td>
<td>679.914/580.789</td>
</tr>
<tr>
<td></td>
<td></td>
<td><em>/</em></td>
</tr>
<tr>
<td></td>
<td></td>
<td><em>/</em></td>
</tr>
<tr>
<td></td>
<td></td>
<td><em>/</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Prob.</th>
<th>Dens</th>
<th>Time, average, secs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>300</td>
<td>18.316/17.179</td>
</tr>
<tr>
<td></td>
<td>90</td>
<td>37.931/34.972</td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>98.690/89.685</td>
</tr>
<tr>
<td></td>
<td>70</td>
<td>413.585/364.480</td>
</tr>
<tr>
<td></td>
<td>60</td>
<td><em>/</em></td>
</tr>
<tr>
<td></td>
<td></td>
<td>85.827/85.096</td>
</tr>
<tr>
<td></td>
<td></td>
<td>173.063/166.996</td>
</tr>
<tr>
<td></td>
<td></td>
<td>679.914/580.789</td>
</tr>
<tr>
<td></td>
<td></td>
<td><em>/</em></td>
</tr>
<tr>
<td></td>
<td></td>
<td><em>/</em></td>
</tr>
<tr>
<td></td>
<td></td>
<td><em>/</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Prob.</th>
<th>Dens</th>
<th>Time, average, secs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>300</td>
<td>18.316/17.179</td>
</tr>
<tr>
<td></td>
<td>90</td>
<td>37.931/34.972</td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>98.690/89.685</td>
</tr>
<tr>
<td></td>
<td>70</td>
<td>413.585/364.480</td>
</tr>
<tr>
<td></td>
<td>60</td>
<td><em>/</em></td>
</tr>
<tr>
<td></td>
<td></td>
<td>85.827/85.096</td>
</tr>
<tr>
<td></td>
<td></td>
<td>173.063/166.996</td>
</tr>
<tr>
<td></td>
<td></td>
<td>679.914/580.789</td>
</tr>
<tr>
<td></td>
<td></td>
<td><em>/</em></td>
</tr>
<tr>
<td></td>
<td></td>
<td><em>/</em></td>
</tr>
<tr>
<td></td>
<td></td>
<td><em>/</em></td>
</tr>
</tbody>
</table>

Table 6.1: Average calculation times (in seconds) against prescribed accuracies of 0% and 5% for instances of the QCP with 100–500 vertices and densities 10%–100% within 10 min.
7. Concluding Remarks

Theorem 4.3a can be considered as a generalization of Theorem 3.1a which is the basis of the DCA(PPAr) and DCA(PPA), respectively. It states that if an interval \([S, T]\) is split into \([S, T - k]\) and \([S + k, T]\), then the maximum value of all differences between the submodular function values \(z\) on levels \(r + 1\) and \(r\) is an upper bound for 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 4.3b can be expounded similarly. These upper bounds are used for implicit “correcting” of the current values of \(z\). In fact, we correct the value of the current accuracy.

We have tested the DCA(PPAr) on cases of the QCP, enabling comparison with results presented in Lee et al. [17]. It is shown that the distance of an optimal solution to the “main diagonal” of the Hasse diagram is a good measure of an instance difficulty of the QCP at least for the DCA(PPAr). This distance increases a little slower than a linear function against the increasing values of the density for any fixed number of vertices (size). The instances with distances from 0% to 20% can be categorized as “hard”, distances from 30% to 60% as “difficult”, and distances from 70% to 100% as “easy”. In all tested instances the average calculation time grows exponentially for decreasing density values for all prescribed accuracies values. This behavior differs from the results of the branch and cut algorithm from Lee et al. [17]. Their calculation times grow when densities increase. This effect is also indicated for all algorithms based on linear programming (see, for example, Barahona et al. [1]; Pardalos and Rodgers [22]; Poljak and Rendl [24]).

Our experiments with different levels \(r\) of the PPAr show that for our instances, the best levels are 3 and 4. In addition, this effect will be more evident when the density of the corresponding instances will be as close as possible to 100%.

Our algorithm can solve instances of the QCP up to 500 vertices on dense graphs within 10 minutes on a standard PC. Therefore it will be interesting to investigate a “composition” of our approach and branch and bound type algorithms based on linear programming for solving instances of the QCP up to 500 vertices for all possible densities.

Recently, Glover et al. [6] reported their computational experiments for binary quadratic programs with adaptive memory tabu search procedures. They assumed that the so called “c” problems with \(n = 200\) 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”. We will try to investigate the computational efficiency of the DCA(PPAr) for large instances of the Quadratic
Cost Partition Problem (QCP) on dense graphs for which the objective will be as close as possible to an objective of the large problems “4f and 5f (see Table 3)” from Glover et al. [6].

References


