Comput Manag Sci (2012) 9:273–286
DOI 10.1007/s10287-012-0140-8

ORIGINAL PAPER

DCA for solving the scheduling of lifting vehicle in an automated port container terminal

Hoai Minh Le · Adnan Yassine · Riadh Moussi

Received: 30 September 2010 / Accepted: 24 January 2012 / Published online: 12 February 2012

Abstract The container was introduced as a universal carrier for various goods in the 1960s and soon became a standard worldwide transportation. The competitiveness of a container seaport is marked by different success factors, particularly the time in port for ships. Operational problems of container terminals is divided into several problems, such as assignment of vessels, loading/unloading and storage of the containers, quay cranes scheduling cite, planning yard cranes cite and assignment of storage containers cite. In this work, the study will focus on piloting yard trucks. Two different types of vehicles can be used, namely automated guided vehicles (AGVs) and lifting vehicles (LVs). An AGV receives a container from a quay crane and transports containers over fixed path. LVs are capable of lifting a container from the ground by itself. The model that we consider is formulated as a mixed integer programming problem, and the difficulty arises when the number of binary variables increases. There are a lot of algorithms designed for mixed integer programming problem such as Branch and Bound method, cutting plane algorithm, ... By using an exact penalty technique we treat this problem as a DC program in the context of continuous optimization. Further, we combine the DCA with the classical Branch and Bound method for finding global solutions.

H. M. Le (✉) · A. Yassine · R. Moussi
Laboratoire de Mathématique Appliquée du Havre,
25 Rue Phillipe Lebon, 76600 Le Havre, France
e-mail: lehoai@univ-metz.fr

R. Moussi
e-mail: riadh.moussi@univ-lehavre.fr

A. Yassine
Institut Supérieur d'Etudes Logistiques (ISEL), Quai Frissard,
BP 1137, 76063 Le Havre Cedex, France
e-mail: yassine@univ-lehavre.fr

Keywords Automated lifting vehicle · Container port terminal · Programmation DC · DCA · Branch and Bound

Mathematics Subject Classification (2000) 90C26 · 90C27 · 90B06

1 Introduction

The container was introduced as a universal carrier for various goods in the 1960s and soon became a standard worldwide transportation. Compared to conventional bulk, the use of containers has several advantages: higher productivity, less product packaging and less damaging. Today over 60% of the world's deep-sea general cargo is transported in containers, especially between economically strong and stable countries. The increasing number of container shipments causes higher demands on the seaport container terminals. Container terminals, especially between geographically close ones, are continuously facing the challenge of strong competition between ports. The competitiveness of a container seaport is marked by different success factors, particularly the time in port for ships and low rates for loading and discharging.

At automated container terminals, containers are transported from one mode of transportation to another. Ship operations consists of the an discharging operation, during which containers are uploaded and stacked in a marshalling yard, and loading operation, during which container are handle in the reverse direction. Operational problems of container terminals is divided into several problems, such as assignment of vessels (Imai et al. 2001), loading/unloading and storage of the containers for a container ship (Imai et al. 2006), quay cranes scheduling (Lee et al. 2008), planning yard cranes (Lee et al. 2007) and assignment of storage containers (Lee et al. 2006). In this work, the study will focus on piloting yard trucks. There are several types of trucks. Two different types of vehicles can be used, namely automated guided vehicles (AGVs) and lifting vehicles (LVs). An AGV receives a container from a quay crane and transports containers over fixed path. In this case, a yard crane is required to take the container off the vehicle. LVs are capable of lifting a container from the ground by itself. To uncouple the unloading process and the transportation process, buffer areas are created where cranes position containers. A LV retrieves a container from a buffer area and transports it to its destination. In this study, we present the terminal of Normandy; Le Havre port. This terminal has a special characteristic which distinguishes him in the other terminals known in the world. It has only two types of handling that are the quay cranes (QCs) and the LVs. A few previous researches have concerned LVs. Iris et al. (2004) have compared the performance of two types of automated vehicles, AGV and ALV, by a simulation study. From this specific study, the authors concluded that, by observing only purchasing costs of equipment, ALVs are a cheaper option than AGVs (38% more AGVs need to be used than ALVs). In Nguyen and Kim (2009), have proposed a mixed zero-one programming model for assigning optimal delivery tasks to LVs. The authors used a heuristic algorithm to solve this problem. In this work, we consider a similar model to the model in Nguyen and Kim (2009).

The model that we consider is formulated as a mixed zero-one programming problem, and the difficulty arises when the number of binary variables increases. There are

a lot of algorithms designed for mixed zero-one programming problem such as Branch and Bound method, cutting plane algorithm and dynamic programming. In this paper, by using an exact penalty technique we treat this problem as a DC program in the context of continuous optimization. Further, we combine the DCA with the classical Branch and Bound method for finding global solutions.

The remainder of the paper is organized as follows. In Sect. 2, we report the description and the formulation of this problem as a mixed zero-one program. Section 3, describes how to reformulate the problem in the continuous form via an exact penalty technique. Section 4, is devoted to the DC programming and DCA for solving the penalty equivalent. The combined DCA & Branch and Bound algorithm is presented in Sect. 5, while the computational results are reported in Sect. 6.

2 Problem definition and mathematical model

2.1 Problem definition

The handling activities performed by QCs are called seaside operations, while those performed by LVs are called landside operations. We define the task of a LV in ship operations as delivering a container from the apron to the yard for a discharging operation or from the yard to the apron for a loading operation. During the discharging operation, a container, picked up by a QC, is put on a buffer space. A LV picks up the container from the buffer and delivers it to the marshalling yard. In the marshalling yard, a LV stacks the container onto an empty slot. The loading operation is performed in the reverse order. The following assumptions are introduced:

- Congestions of LVs are not considered.
- All LVs are same and transfer one container at a time.
- Pick up and release time of a contrainer by QC/LV can be neglected.

2.2 Mathematical model

- Let q_i^k be an event representing the moment that QC k transfers the ith container. When the ith operation of QC k is a loading operation, q_i^k corresponds to the beginning of collecting of a container from the ground. When the operation of QC k is an unloading operation, q_i^k corresponds to the beginning of the release of a container.
- y_i^k: the event time of q_i^k.
- s_i^k: the earliest event time of q_i^k (represent the date when the QC is free). The delay of an operation occurs when the LV corresponding does not arrive at the required moment.
- Let a_i^k be an event representing the moment that a LV transfers the ith container. When the ith operation of QC k is a loading operation, a_i^k corresponds to the release of the ith container by a LV under QC k. When the operation of QC k is an unloading operation, a_i^k corresponds to the pickup of the ith container by a LV from the pickup point of QC k.

- z_i^k: the event time of a_i^k.
- We add two fictive QCs: O and F. One denotes: a_j^O(resp. a_j^F) the starting event of LV j(resp. the stopping event of LV j).
- $l(a_i^k)$: the position of the event a_i^k. $l(a_i^O)$ represents the initial position of LV i and $l(a_i^f)$ represents the final position of LV i.
- t_{ki}^{lj}: the pure travel time from $l(a_i^k)$ to $l(a_j^l)$.
- c_{ki}^{lj}: the time required for a LV to be ready for a_j^l after a_i^k. For example, if both a_i^k and a_j^l are related to loading operations, then the starting event for evaluating c_{ki}^{lj} is the release of the ith container by LV. c_{ki}^{lj} includes the travel time from an apron to the location of the next container (the jth container of QC l) in the marshalling yard, the pickup time of the container by LV, and the travel time of the LV to QC l.
- V: the set of LVs.
- K: the set of QCs. $K' = \{0\} \cup K$; $K'' = \{F\} \cup K$.
- m_k: the set of tasks of every QC K.
- S: the set of a_j^0.
- D: the set of a_j^f.
- T: the set of a_k^i.
- L^k : the set of loading tasks of QC k.
- U^k : the set of unloading tasks of QC k.

A feasible dispatching decision of forwarding is an assignment one to one between all the events of $S \cup T$ and $D \cup T$. Let $k' = \{0\} \cup k$, $K'' = \{F\} \cup k$ and x_{ki}^{lj} be a decision variable x_{ki}^{lj}. $x_{ki}^{lj} = 1$ if a_i^k is assigned to a_j^l for $k \in K'$ and $l \in K''$(implies that the LV, which has just delivered the ith container of QC k, will deliver the jth container of QC l), 0 otherwise. One denotes the travel cost per unit time of a LV by α and the penalty cost per unit time for the delay in the completion time by β. We assume that $\alpha \ll \beta$.

We minimize at same time the total travel time of LVs and the total delays of QCs. Since $\alpha \ll \beta$, the total delays of QCs is much more important than the total travel time of LVs. Objective function is defined as follows:

$$\min f(x, y) = \alpha \sum_{k \in K'} \sum_{i=1}^{m_k} \sum_{l \in K''} \sum_{j=1}^{m_l} t_{ki}^{lj} x_{ki}^{lj} + \beta \sum_{k \in K} (y_{m_k}^k - s_{m_k}^k) \tag{1}$$

The constraints are summarized as follows:

1. One-to-one assignment between $S \cup T$ and $D \cup T$:

$$\begin{aligned} \sum_{l \in K''} \sum_{j=1}^{m_l} x_{ki}^{lj} &= 1 \quad \forall k \in K'; \quad i = 1 \ldots m_k \\ \sum_{k \in K'} \sum_{i=1}^{m_k} x_{ki}^{lj} &= 1 \quad \forall l \in K''; \quad j = 1 \ldots m_l \end{aligned} \tag{2}$$

2. Two events which are carried out in a consecutive way by the same LV must be set apart at least the time necessary by a LV to travel and to transfer its load between the two events.

$$z_j^l - (z_i^k + c_{ki}^{lj}) \geq M(x_{ki}^{lj} - 1) \quad \forall k \in K', \quad l \in K, \quad i = 1 \ldots m_k, \quad j = 1 \ldots m_l \tag{3}$$

with M, a big positive number.

3. Two events which are been useful by the same QC must be set apart at least the time necessary with the QC to carry out all the movements between the two events.

$$y_{i+1}^k - y_i^k \geq s_{i+1}^k - s_i^k \quad \forall k \in K, \quad i = 1 \ldots m_k \tag{4}$$

4. The actual event time is always equal to or greater than the date earliest possible

$$y_i^k \geq s_i^k \quad \forall k \in K, \quad i = 1 \ldots m_k \tag{5}$$

Finally we get the following optimization problem:

$$(P) \begin{cases} \min f(x, y) = \alpha \sum_{k \in K'} \sum_{i=1}^{m_k} \sum_{l \in K''} \sum_{j=1}^{m_l} t_{ki}^{lj} x_{ki}^{lj} + \beta \sum_{k \in K} (y_{m_k}^k - s_{m_k}^k) \\ \text{Subject to:} \\ (2) - (5) \\ x_{ki}^{lj} \in \{0, 1\} \quad \forall k \in K', \quad l \in K'', \quad i = 1 \ldots m_k, \quad j = 1 \ldots m_l \end{cases} \tag{6}$$

We are then facing the minimization of a linear function with mixed zero-one variables under linear constraints.

3 Reformulation

In this section, using the well known results concerning the exact penalty we will formulate (P) in the form of concave minimization programming. Consider now the mixed zero-one linear program in a general form:

$$\alpha = \min \left\{ c^T x + d^T y : (x, y) \in D, x \in \{0, 1\}^m \right\} \tag{MIP}$$

where $x \in \{0, 1\}^m$ and $y \in \mathbb{R}^n$ are binary and continuous variables respectively, and D is a nonempty bounded polyhedral convex set in $\mathbb{R}^m \times \mathbb{R}^n$ defined by a finite number of linear constraints.

Let us consider the function p defined by

$$p(x, y) = \theta(x) := \sum_{i=1}^{m} \min(x_i, 1 - x_i).$$

Set $K := \{(x, y) \in D : x \in [0, 1]^m\}$. Clearly, p is concave and finite on K, $p(x, y) \geq 0$ for all $(x, y) \in K$, and

$$\{(x, y) \in D, \quad x \in \{0, 1\}^m\} = \{(x, y) \in K, \quad p(x, y) \leq 0\}.$$

Hence Problem (MIP) can be rewritten as

$$\alpha = \min : \{c^T x + d^T y : (x, y) \in K, \quad p(x, y) \leq 0\}. \tag{7}$$

From Theorem 1 below we get, for a sufficiently large value of t ($t \geq t_0$), the equivalent concave minimization problem to (MIP):

$$\min : \{c^T x + d^T y + tp(x, y) : (x, y) \in K\}. \tag{8}$$

Theorem 1 (Theorem 1, Le Thi et al. 1999) *Let K be a nonempty bounded polyhedral convex set, f be a finite concave function on K and p be a finite nonnegative concave function on K. Then there exists $t_0 \geq 0$ such that the following problems have the same solution set and the same optimal value:*

$$\alpha(t) = \inf\{f(x) + tp(x) : x \in K\} \tag{P_t}$$

$$\alpha = \inf\{f(x) : x \in K, p(x) \leq 0\}. \tag{P}$$

More precisely if the vertex set of K, denoted by $V(K)$, is contained in $\{x \in K, p(x) \leq 0\}$, then $t_0 = 0$, otherwise $t_0 = \min\left\{\frac{f(x)-\alpha(0)}{S} : x \in K, p(x) \leq 0\right\}$, where $S := \min\{p(x) : x \in V(K), p(x) > 0\} > 0$.

Now let $x \in \mathbb{R}^m$ be the binary variables and $y \in \mathbb{R}^n$ be the continuous variables in Problem (P). Clearly, the set D of feasible points (x, y) determined by the system of the constraints (2)–(5) is a nonempty, bounded polyhedral convex set in $\mathbb{R}^n \times \mathbb{R}^n$. Problem ($P$) can be expressed in the form of (MIP) and so the above result is available for (P).

4 A DCA scheme for solving Problem (MIP)

4.1 DC Programming

In this section we investigate a DC programming approach for solving (8). DC programming and DCA constitute the backbone of smooth/nonsmooth nonconvex programming and global optimization. They address the problem of minimizing a function f which is the difference of two convex functions on the whole space $\mathbb{R}^p$ or on a convex set $C \subset \mathbb{R}^p$. Generally speaking, a DC program is an optimisation problem of the

form:

$$\alpha = \inf\{f(x) := g(x) - h(x) : x \in \mathbb{R}^p\} \qquad (P_{dc})$$

where g, h are lower semi-continuous proper convex functions on $\mathbb{R}^p$. Such a function f is called a DC function, and g–h a DC decomposition of f while g and h are the DC components of f. The convex constraint $x \in C$ can be incorporated in the objective function of (P_{dc}) by using the indicator function on C denoted by χ_C which is defined by $\chi_C(x) = 0$ if $x \in C$, and $+\infty$ otherwise:

$$\inf\{f(x) := g(x) - h(x) : x \in C\} = \inf\{\chi_C(x) + g(x) - h(x) : x \in \mathbb{R}^p\}.$$

Let

$$g^*(y) := \sup\{\langle x, y\rangle - g(x) : x \in \mathbb{R}^p\}$$

be the conjugate function of a convex function g. Then, the following program is called the dual program of (P_{dc}):

$$\alpha_D = \inf\{h^*(y) - g^*(y) : y \in \mathbb{R}^p\}. \qquad (D_{dc})$$

One can prove that $\alpha = \alpha_D$, and there is the perfect symmetry between primal and dual DC programs: the dual to (D_{dc}) is exactly (P_{dc}). For a convex function θ, the subdifferential of θ at $x_0 \in \text{dom}\,\theta := \{x \in \mathbb{R}^p : \theta(x_0) < +\infty\}$, denoted by $\partial\theta(x_0)$, is defined by

$$\partial\theta(x_0) := \{y \in \mathbb{R}^n : \theta(x) \geq \theta(x_0) + \langle x - x_0, y\rangle, \forall x \in \mathbb{R}^p\}. \qquad (9)$$

The subdifferential $\partial\theta(x_0)$ generalizes the derivative in the sense that θ is differentiable at x_0 if and only if $\partial\theta(x_0) \equiv \{\nabla_x\theta(x_0)\}$.

DCA is based on the local optimality conditions of (P_{dc}), namely

$$\partial h(x^*) \cap \partial g(x^*) \neq \emptyset \qquad (10)$$

(such a point x^* is called a *critical point* of g–h), and

$$\emptyset \neq \partial h(x^*) \subset \partial g(x^*). \qquad (11)$$

Note that (11) is a necessary local optimality condition for (P_{dc}). For many classes of the DC program, it is also a sufficient optimality condition (see Le Thi and Pham (2005, 1997)).

The idea of DCA is simple : each iteration l of DCA approximates the concave part $-h$ by its affine majorization (that corresponds to taking $y^l \in \partial h(x^l)$) and minimizes the resulting convex function (that is equivalent to determining a point $x^{l+1} \in \partial g^*(y^l)$).

DCA scheme
Initialization: Let $x^0 \in \mathbb{R}^p$ be a best guess, $0 \leftarrow k$.
Repeat

- Calculate $y^l \in \partial h(x^l)$
- Calculate $x^{l+1} \in \arg\min\{g(x) - h(x^l) - \langle x - x^l, y^l\rangle : x \in \mathbb{R}^p\}$ (P_l)
- $l + 1 \leftarrow l$

Until convergence of the values x^l.
Note that (P_l) is a convex optimisation problem and in so far "easy" to solve. Convergence properties of DCA and its theoretical basis can be found in (Le Thi and Pham 2005; Pham 1988; Le Thi and Pham 1997).

4.2 DCA for solving Problem (8)

We first prove that (8) is a DC program and then present the DCA applied to the resulting DC program.

Denote by χ_K the indicator function on K, i.e., $\chi_K(x, y) = 0$ if $(x, y) \in K$, $+\infty$ otherwise. Let g and h be functions defined by

$$g(x, y) = \chi_K(x, y) \quad \text{and} \quad h(x, y) = -c^T x - d^T y - t \sum_{i=1}^{m} min(x_i, 1 - x_i). \tag{12}$$

Hence g and h are convex functions, and so Problem (8) is a DC program in the form

$$\min\{g(x, y) - h(x, y) \ : \ (x, y) \in \mathbb{R}^m \times \mathbb{R}^n\}. \tag{13}$$

By the very definition of h, a subgradient $(u, v) \in \partial h(x, y)$ can be chosen as follows:

$$(u, v) \in \partial h(x, y) \Leftarrow u = -c, \quad v = (v_i)_i, \text{ with } v_i = \begin{cases} -1 - d_i & \text{if } y_i \geq 0.5 \\ 1 - d_i & \text{otherwise.} \end{cases} \tag{14}$$

Algorithm 1 (DCA applied to (8))

1. Let $(x^0, y^0) \in \mathbb{R}^m \times \mathbb{R}^n$. Set $k = 0$ and let ϵ_1, ϵ_2 be sufficiently small positive numbers.
2. Compute $(u^k, v^k) \in \partial h(x^k, y^k)$ via (14).
3. Solve the linear program:

$$\min\left\{-\langle (u^k, v^k), (x, y) : (x, y) \in K\right\} \tag{15}$$

to obtain (x^{k+1}, y^{k+1}).

4. If either $\left\|(x^{k+1}, y^{k+1}) - (x^k, y^k)\right\| \leq \epsilon_1(\left\|(x^k, y^k)\right\| + 1)$ or

$$\left|\langle c, x^{k+1} - x^k\rangle\langle d, y^{k+1} - y^k\rangle + t\sum_{i=1}^{m}\left(\min(x_i^{k+1}, 1 - x_i^{k+1}) - \min(x_i^k, 1 - x_i^k)\right)\right|$$
$$\leq \epsilon_2\left(\left|\langle c, x^k\rangle + \langle d, y^k\rangle + t\sum_{i=1}^{m}\min(x_i^k, 1 - x_i^k)\right| + 1\right)$$

then stop, (x^k, y^k) is the computed solution, otherwise, set $k = k + 1$ and go to Step 2.

The convergence of Algorithm 1 can be summarized in the next theorem whose proof is essentially based on the DCA's Convergence Theorem for a polyhedral DC program (Le Thi and Pham 2005; Pham 1988; Le Thi and Pham 2002).

Theorem 2 (Theorem 2, Convergence properties of Algorithm 1)

(i) *Algorithm* **1** *generates a sequence* $\{(x^k, y^k)\}$ *contained in* $V(K)$ *such that the sequence* $\{g(x^k, y^k) - h(x^k, y^k)\}$ *is decreasing.*

(ii) *For a sufficiently large value of* t*, if at iteration* r *we have* $x^r \in \{0, 1\}^m$*, then* $x^k \in \{0, 1\}^m$ *for all* $k \geq r$.

(iii) *The sequence* $\{(x^k, y^k)\}$ *converges to* $(x^*, y^*) \in V(K)$ *after a finite number of iterations. The point* (x^*, y^*) *is a critical point of Problem* (8)*. Moreover if* $x_i^* \neq \frac{1}{2}$ *for* $i = 1, \ldots, m$*, then* (x^*, y^*) *is a local minimizer to Problem* (8)*.*

Proof (i) is a consequence of DCA's Convergence Theorem for a general DC program (see Le Thi and Pham 2001, 2005; Pham 1988; Pham and Le Thi 1997).

(ii) Let $t > t_1 := \max\left\{\frac{\langle c,x\rangle + \langle d,y\rangle - \alpha}{\beta} : (x, y) \in V(K), \theta(x) \leq 0\right\}$, where $\alpha := \min\{\langle c, x\rangle + \langle d, y\rangle : (x, y) \in V(K)\}$ and $\beta := \min\{\theta(x) : (x, y) \in V(K)\}$. Let $\{(x^k, y^k)\} \subset V(K)$ $(k \geq 1)$ be generated by Algorithm 1. If $V(K) \subset \{0, 1\}^n$, then the assertion is trivial. Otherwise, let $x^r \in \{0, 1\}^n$ and $(x^{r+1}, y^{r+1}) \in V(K)$ be an optimal solution of the linear program (15). Then from (i) of this theorem we have

$$\langle c, x^{r+1}\rangle + \langle d, y^{r+1}\rangle + t\theta(x^{r+1}) \leq \langle c, x^r\rangle + \langle d, y^r\rangle + t\theta(x^r).$$

Since $\theta(x^r) = 0$, it follows

$$t\theta(x^{r+1}) \leq \langle c, x^r\rangle + \langle d, y^r\rangle - \langle c, x^{r+1}\rangle - \langle d, y^{r+1}\rangle \leq \langle c, x^r\rangle + \langle d, y^r\rangle - \alpha.$$

If $\theta(y^{r+1}) > 0$, then

$$t \leq \frac{\langle c, x^r\rangle + \langle d, y^r\rangle - \langle c, x^{r+1}\rangle - \langle d, y^{r+1}}{\theta(x^{r+1})} \leq \frac{\langle c, x^r\rangle + \langle d, y^r\rangle - \alpha}{\beta} \leq t_1$$

which contradicts the fact that $t > t_1$.

(iii) Problem (8) with the DC decomposition (12) is a polyhedral DC program of the form (4.1) since the second DC component h is a polyhedral convex function. (In fact the first DC component g is polyhedral convex too). So DCA applied to (4.1) has a finite convergence (Le Thi and Pham 2005; Pham 1988; Le Thi and Pham 2002). According to the DCA's convergence theorem, for any DC program, the solution computed by DCA is a critical point of g–h, i.e.,

$$\partial g(x^*, y^*) \cap \partial h(x^*, y^*) \neq \emptyset. \tag{16}$$

If $x_i^* \neq 1/2, \forall i = 1, \ldots, m$, then h is differentiable at (x^*, y^*) and then the condition (16) becomes $\partial h(x^*, y^*) \subset \partial g(x^*, y^*)$. This subdifferential inclusion (which is a necessary condition of local optimality in DC programming) is also sufficient in the case of a polyhedral DC program whose second DC component h is a polyhedral convex function (Le Thi and Pham 2005; Pham 1988; Le Thi and Pham 2002). The proof is then complete. □

We remark that *Algorithm 1* converges to a local solution after a **finite number of iterations** and it consists in **solving a linear program at each iteration**. Moreover, although the **DCA works on a continuous domain, it provides an integer solution**.

5 A combined DCA-Branch and Bound algorithm for solving Problem (MIP)

For globally solving the problem we combine the DCA with the classical Branch and Bound method applied to (MIP). The linear relaxation is used for computing lower bounds while the upper bounds are determined by applying DCA to (8). Our combined algorithm can be summarized as follows:

DCA-BB.
Let $R_0 := [0, 1]^m$. Set $\gamma_0 := +\infty$, $\beta_0 := -\infty$, $restart := true$, $\mathscr{R} := \{R_0\}$, and $k = 0$. Let ϵ be sufficiently small positive number.

1. Let R_k be the rectangle such that $\beta_k = \beta(R_k) = \min\{\beta(R) : R \in \mathscr{R}\}$.
   Bisect R_k into two subrectangles R_{k_0} and R_{k_1} via the index $j*$

$$R_{k_i} = \{x \in R_k : \quad x_{j*} = i, \quad i = 0, 1\}$$

2. Compute lower bounds β_{k_i} $(i = 1, 2)$ by solving the linear relaxation problems corresponding to the set R_{k_i}.
3. If $(restart = true)$ then update γ_k, the best upper bound of the optimal value of (MIP) by applying DCA to Problem (8) from a suitable starting point discovered in Step 2.
4. If $\mathscr{R} =$ (i.e. $\gamma_k - \beta_k \leq \epsilon$), then STOP, the optimal solution is (x^k, y^k) that verify $c^T x^k + d^T y^k = \gamma_k$, otherwise update

$$\mathscr{R} \leftarrow \mathscr{R} \cup \{R_{k_i} : \beta(R_{k_i}) < \gamma_k - \epsilon, i = 0, 1\} \setminus R_k$$

and go to Step 1.

Our combined DCA-BB differs from the classical Branch and Bound scheme by Step 3 in which DCA is investigated. In fact, the Branch and Bound algorithm is introduced to find a good starting point for DCA and check the globality of DCA. *restart* is a boolean variable which takes value *true* if we decide to restart DCA. The question *when DCA is restarted* is interesting from numerical points of view and it will be studied in Sect. 5.2. As in several DC programs, DCA provides a global solution to (8) (and so is to (MIP)) from a *good* starting point. This question will be studied in Sect. 5.1.

5.1 How to find a good starting point for DCA?

From Theorem 2 we see that, starting with a feasible solution to (MIP) DCA provides a better feasible solution, although it works on a continuous feasible set of (8). It is so important to find a good feasible point to (MIP) for restarting DCA. We can restart DCA from the best feasible solution to (MIP) that is discovered while computing lower bounds. This is motivated by a similar and efficient way introduced in the combined DCA-Branch and Bound algorithm for nonconvex quadratic programming (Le Thi and Pham 2001).

On the other hand, for obtaining rapidly a good feasible point to (MIP) we also investigate a fast procedure to compute an optimal solution of the concave quadratic program

$$0 = \min\left\{\sum_{i=1}^{m} x_i(1 - x_i) : (x, y) \in K\right\}. \tag{17}$$

That is the DCA developed in (Le Thi and Pham 2001).

In our algorithm, a suitable starting point is computed by choosing one of the two procedures according to the current situation.

5.2 When do we restart DCA?

During the branch and bound process we restart DCA in two cases:

- when a feasible solution to (MIP) which improves the best current upper bound is found. In such a case, the starting point of DCA is the just mentioned feasible solution to (MIP).
- when the number of the 0–1 components of the binary variables (denoted $N_{x^{R_k}}$) of the solution (x^{R_k}, y^{R_k}) to the corresponding linear relaxation problem is sufficiently large, namely $N_{x^{R_k}} \geq m/2$. The starting point of DCA in this case is the solution of Problem (17).

5.3 Algorithm 2 (DCA-BB)

We will now describe the combined DCA-Branch and Bound algorithm for solving Problem (MIP).

1. Let $R_0 := [0, 1]^m$.
2. Solve the linear relaxation problem of (MIP) to obtain an optimal solution (x^{R_0}, y^{R_0}) and the first lower bound $\beta_0 := \beta(R_0)$.
3. Solve (8) by DCA from the starting point (x^{R_0}, y^{R_0}) to obtain $(x_t^{R_0}, y_t^{R_0})$.
   If $(x_t^{R_0}, y_t^{R_0})$ is feasible to (MIP) then set $\gamma_0 := c^T x_t^{R_0}$ and set $(x^0, y^0) := (x_t^{R_0}, y_t^{R_0})$.
   else set $\gamma_0 := +\infty$.
4. If $(\gamma_0 - \beta_0) \leq \epsilon|\gamma_0|$ then (x^0, y^0) is an ϵ-optimal solution of (MIP)
   else set $\mathscr{R} \leftarrow \{R_0\}, k \leftarrow 0$.
5. While $stop = false$ do
   (a) Select a rectangle R_k such that $\beta_k = \beta(R_k) = \min\{\beta(R) : R \in \mathscr{R}\}$.
   (b) Bisect R_k into two subrectangles R_{k_0} and R_{k_1} via the index $j*$

$$R_{k_i} = \{x \in R_k : x_{j*} = i, \quad i = 0, 1\}$$

   (c) Solve the subproblems (P_{k_i}) to obtain $\beta(R_{k_i})$ and $(x^{R_{k_i}}, y^{R_{k_i}})$:

$$\beta(R_{k_i}) := \min\left\{c^T x + d^T y : (x, y) \in K, \quad x \in R_{k_i}\right\} \quad (i = 0, 1). \tag{P_{k_i}}$$

   If $(x^{R_{k_i}}, y^{R_{k_i}})$ is the best feasible solution to (MIP) then update γ_k and the best feasible solution (x^k, y^k) by applying DCA to (8) from $(x^{R_{k_i}}, y^{R_{k_i}})$.
   Else if $(N_{y^{R_{k_i}}} \geq m/2)$ then solve (17) by DCA to obtain $(x_t^{R_{k_i}}, y_t^{R_{k_i}})$.
   Apply DCA to (8) from $(x_t^{R_{k_i}}, y_t^{R_{k_i}})$. Update γ_k and the best feasible point (x^k, y^k).
   Endif
   (d) Set $\mathscr{R} \leftarrow \mathscr{R} \cup \{R_{k_i} : \beta(R_{k_i}) < \gamma_k - \epsilon, i = 0, 1\} \setminus R_k$
   (e) If $\mathscr{R} = \emptyset$ then STOP, (x^k, y^k) is ϵ-optimal solution, else $k \leftarrow k + 1$.
   Endwhile

6 Numerical experiments

In this section, in order to evaluate the performance of the proposed algorithm, we randomly generated a number of problems by changing the number of QCs, the number of LVs and the number of containers assigned to each QC. Table 1 displays a few line of an example of a sequence list applied in ship operations for a QC.

The algorithm was implemented in C++ and run on a Intel Core 2CPU 1.86Ghz of 2GB RAM. For solving linear programs, CPLEX 9.1 was used. We used also CPLEX 9.1 for solving the original problem (P). In this experiment, we compare the efficiency of three algorithms: our combined $BB - DCA$, the classical Branch and Bound algorithm and CPLEX 9.1. For the tolerance, $\epsilon \leq 5.10^{-2}$ is chosen for all the methods. The time limit of CPLEX 9.1 is limited to 2 h. We use following notations:

- n: number of quay cranes.
- m: number of vehicles.

Table 1 Exemple of a sequence list

Container ID	Type[a]	Ship location[b]	Yard location[c]	Earliest event time
1	L	(12, 2)	(65, 3, 1)	0
3	U	(24, 3)	(64, 10, 2)	210
6	L	(14, 1)	(61, 12, 3)	210
7	U	(12, 1)	(67, 14, 2)	420
10	L	(13, 2)	(12, 18, 1)	420
12	U	(15, 2)	(132, 2, 3)	630

[a] *L* loading, *U* unloading
[b] (N row, N tier)
[c] (N block, N row, N tier)

Table 2 Comparaison of BB-DCA, BB and CPLEX 9.1

N	**Size**		**BB**		**BB-DCA**		**CPLEX 9.1**	
	n	*m*	Gap	Time	Gap	Time	Gap	Time
1	2	10	**0.0033**	425	**0.0033**	**310**	**0.0033**	367
2	3	15	0.0017	721	0.0010	**412**	**0.0005**	745
3	2	15	0.0025	512	0.0027	**365**	**0.0023**	376
4	3	20	0.0034	861	**0.0013**	**432**	0.0018	812
5	2	10	**0.0012**	336	**0.0012**	212	**0.0012**	**187**
6	2	15	0.0006	442	**0.0005**	**212**	**0.0005**	218
7	3	15	0.0034	1,056	0.0021	**632**	**0.0011**	645
8	3	20	0.0134	1,103	**0.0087**	**871**	0.0101	1,024
9	3	10	0.0125	774	0.0102	**489**	**0.0078**	891
10	3	20	0.0131	934	0.0111	**645**	**0.0093**	1,020
Average			0.0055	736	0.0040	**458**	**0.0038**	628

Bold values are the best results

- UB : the last upper bound of each algorithm.
- LB : the last lower bound of each algorithm.
- $Gap = (\text{UB} - \text{LB})/(1 + \text{UB})$.
- $Time$: CPU time for each algorithm in seconds.

The comparative results between three algorithms are reported in Table 2. 10 sizes of test are considered (10 instances per size).

From numerical results, we observe that:

- In all test size, BB-DCA gives better results than classical BB, not only the quality of solution but also the CPU time. DCA is inexpensive and can so handle problems with large number of binary variables. The superiority of DCA-BB relative to the Branch and Bound algorithm increases when the number of binary variables increases. DCA-BB is fast for large-scale problems while Branch and Bound algorithm is quite slow or it cannot solve some problems in reasonable times.
- CPLEX 9.1's quality of solution is slightly better than BB-DCA's one but BB-DCA is faster than CPLEX 9.1 in most of cases.

References

An LHA, Tao DP (1997) Solving a class of linearly constrained indefinite quadratic problems by DC algorithms. J Global Optimiz 11(3):253–285

An LTH, Tao PD (2001) A continuous approach for globally solving linearly constrained quadratic zero—once programming problems. Optimization 50:93–120

An LTH, Tao PD (2002) DC programming: theory, algorithms and applications. In: The State of the Art (28 pages), Proceedings (containing the refereed contributed papers) of The First International Workshop on Global Constrained Optimization and Constraint Satisfaction (Cocos' 02), Valbonne-Sophia Antipolis, France, October 2–4

An LTH, Tao DP (2005) The DC (Difference of convex functions) programming and DCA revisited with DC models of real world nonconvex optimization problems. Ann Oper Res 133:23–46

An LTH, Tao DP, Muu LD (1999) Exact penalty in D. C. programming. Vietnam J Math 27(2):169–178

Imai A, Nishimura E, Papadimitriou S (2001) The dynamic berth allocation problem for a container port. Transport Res Part B 35(4):401–417

Imai A, Sasaki K, Nishimura E, Papadimitriou S (2006) Multi-objective simultaneous stowage and load planning for a container ship with container rehandle in yard stacks. Er J Oper Res 171(2):373–389

Kim KH, Jong WB (2004) A look-ahead dispatching method for automated guided vehicles in automated port container terminals. Transp Sci 38(2):224–234

Lee DH, Cao Z, Meng Q (2007) Scheduling of two-transtainer systems for loading outbound containers in port container terminals with simulated annealing algorithm. Int J Product Econom 107(1):115–124

Lee LH, Chew EP, Tan KC, Han Y (2006) An optimization model for storage yard management in transshipment hubs. OR Spectrum 28(4):539–556

Lee DH, Wang HQ, Miao LX (2008) Quay crane scheduling with non-interference constraints in port container terminals. Transport Res Part E 44(1):124–135

Nguyen VD, Kim KH (2009) A dispatching method for automated lifting vehicles in automated port container terminals. Comput Ind Eng 56(3):1002–1020

Tao PD (1988) Duality in DC (Difference of convex functions) optimization, subgradient methods, trends in mathematical optimization. Int Ser Number Math Birkhauser 84:277–293

Tao PD, An LTH (1997) Convex analysis approach to DC programming: theory, algorithms and applications. Acta Math Vietnamica, dedicated to Professor Hoang Tuy on the occasion of his 70th birthday 22(1):289–355

Vis IFA, Harika I (2004) Comparison of vehicle types at an automated container terminal. OR Spectrum 26(1):117–143