On solving the Cross-dock Door Assignment Problem, CDAP

Laureano F. Escudero[a] and M. Araceli Garín[b] and Aitziber Unzueta[c]

[a]Statistics and Operations Research area. Universidad Rey Juan Carlos, URJC, Móstoles (Madrid), Spain;
[b]Quantitative Methods department. Universidad del País Vasco, UPV/EHU, Bilbao (Bizkaia), Spain;
[c]Applied Mathematics department. Universidad del País Vasco, UPV/EHU, Bilbao (Bizkaia), Spain

ARTICLE HISTORY
Compiled September 9, 2022

ABSTRACT
A class of strong lower bounds on the solution value of a Linearized Integer Programming (LIP) reformulation is introduced for the binary quadratic optimization model to assign origin and destination nodes to strip and stack doors, resp., in a cross-dock infrastructure, whose goal is to minimize the transportation cost of the commodities to be handled at the cross-dock. Strip- and stack-related decomposition submodels are developed by taking benefit of the Integer Linearization Property that appears in enlarged Generalized Assignment Problems (GAPs) as embedded in the original model. It is proved that their further tightening defines two new equivalent reformulations of the LIP model. Additionally, a linear search heuristic is provided for obtaining feasible solutions by exploiting the GAP structures. We present an extensive computational study on a broad set of instances that shows that the proposed joint scheme for lower bounding and feasible solution providing is very efficient. The scheme obtains an optimal solution for small and middle size instances taken from the literature, although straightforward use of a state-of-the-art MILP optimizer requires less computing time for the small ones. Moreover, the proposal outperforms the optimizer when solving the largest instances.

KEYWORDS
Supply chain distribution, cross-docking, decomposition, decoupling, heuristic algorithms.

1. Problem description and binary quadratic modeling

1.1. *Problem description*

Given a network with commodities to deliver from origin nodes to destination ones, the cross-docking problem basically consists of using a entity to serve as a consolidation point for helping on supply chain distribution. The origin nodes can deliver the commodities in the cross-dock, so that, after being clas-

[a]Email: laureano.escudero@urjc.es
[b]Email: mariaaraceli.garin@ehu.eus
[c]Email: aitziber.unzueta@ehu.eus

sified by type and destination, they can be transported (usually, in smaller quantities) to the destination nodes. A cross-dock has a set of inbound infrastructures (say, doors, so-named strip ones, vehicles, etc.), and a set of outbound ones (say, doors, so-named stack doors, vehicles, etc.), each of them with a capacity for commodity handling during a given time period. The goal consists of assigning the origin and destination nodes to the inbound and outbound infrastructures, resp., so that the commodity handling and transportation cost in the cross-docking is minimized.

As a matter of fact, industry detected the importance of cross-docking problem earlier than academia. It seems that Wal-Mart and some others in the retailer industry, see Stalk, Evans, and Shulman (1992), were pioneering its applicability in the late 80s, jointly with UPS, in particular, for its parcel delivering system, see Forger (1995). Office Depot is another company that took benefit of cross-docking in the earlier years, see Ross (1997). Cross-docking implementations have started to be frequent in fast-moving consumer-goods, manufacturing, automation and, in general, commodity distribution, say, from big companies to small ones; for specific implementations, see e.g., Kinnear (1997); Witt (1998); Napolitano (2011). Serrano, Delorme, and Dolgui (2021) present an application of cross-docking dynamic operations at an international logistic network of Renault. See also this type of cross-docking application in Coindreau et al. (2021) for another large car manufacturer. Both works present different MILP models for minimizing the commodity inventory penalization when the truck origin node flow is not synchronized, in the same time period, with the container destination nodes. For good overviews on cross-docking applications, see Ladier and Alpan (2006); VanBelle, Valckenaers, and Cattrysse (2012); Kiani Mavi et al. (2020); Akkerman et al. (2022).

The scientific cross-docking problem solving is a difficult task. It was not until the first decade of the current century when a strong advancement in combinatorial theory and algorithmic has allowed to develop tools for problem solving. In particular, Cross-dock Door Assignment Problem (CDAP), as one of the key structures of cross-docking, can formally be stated as follows. Let a set of origin nodes M, a set of destination nodes N, a set of strip doors I, and a set of stack doors J. Each strip/stack door may serve more than one origin/destination node satisfying the doors' capacity constraints. On the other hand, each origin/destination must be allocated to one strip/stack door. Additionally, let the following notation for the data to be used throughout the work, see Zhu et al. (2009); Guignard et al. (2012); Nassief, Contreras, and As'ad (2016); Gelareh et al. (2020).

d_{ij}, commodity handling and transportation unit cost (so-named distance, in the literature) from strip door i to stack door j, for $i \in I$, $j \in J$.

w_{mn}, commodity volume from origin node m to be delivered at destination node n, for $m \in M$, $n \in N$, so that $d_{ij}w_{mn}$ is the commodity handling and transportation total cost from node i to node j using the strip door i and stack door j.

s_m and r_n, commodity total volume arriving from origin node m, for $m \in M$, and delivering to node n, for $n \in N$, resp., such that $s_m = \sum_{n \in N} w_{mn}$ and $r_n = \sum_{m \in M} w_{mn}$.

S_i and R_j, commodity handling capacity of strip door i, for $i \in I$, and stack door j, for $j \in J$, resp.

The binary variables are as follows:

$x_{mi} = 1$, its value 1 means that origin m is assigned to strip door i; otherwise, 0.
$y_{nj} = 1$, its value 1 means that destination n is assigned to stack door j; otherwise, 0.

1.2. *Binary Quadratic modeling*

The assignment of origin and destination nodes to strip and stack doors, resp., can be represented by Generalized Assignment Problems (GAPs), to be expressed as

$$\sum_{m \in M} s_m x_{mi} \leq S_i \qquad \forall i \in I \tag{1a}$$

$$\sum_{i \in I} x_{mi} = 1 \qquad \forall m \in M \tag{1b}$$

$$x_{mi} \in \{0, 1\} \qquad \forall m \in M, i \in I \tag{1c}$$

and

$$\sum_{n \in N} r_n y_{nj} \leq R_j \qquad \forall j \in J \tag{2a}$$

$$\sum_{j \in J} y_{nj} = 1 \qquad \forall n \in N \tag{2b}$$

$$y_{nj} \in \{0, 1\} \qquad \forall n \in N, j \in J. \tag{2c}$$

The objective function that links the GAPs is a binary quadratic one on $x_{mi} y_{nj}$. So, model CDAP can be expressed

$$z^* = \min \sum_{m \in M} \sum_{i \in I} \sum_{n \in N} \sum_{j \in J} d_{ij} w_{mn} x_{mi} y_{nj} \tag{3a}$$

$$\text{s. to constraint systems GAP } (x, S) \text{ (1) and GAP } (y, R) \text{ (2).} \tag{3b}$$

The model belongs to the category so-named BQP (Binary Quadratic Problem) as a special mixed integer nonlinear problem. There are several alternatives for BQP solving, some of them have already been implemented in state-of-the-art solvers as CPLEX (the basic solver used in this work); it considers the so-named Fortet inequalities, see Fortet (1960). Other alternatives use reformulations as the so-named *convexification* of the quadratic objective function, see Billionnet, Elloumi, and Plateau (2009). A recent alternative is based on the Dantzig-Wolfe Reformulation, where the constraint subsets in the model are replaced with the convex hull of its extreme points, see Cesselli, Létocart, and Traversi (2022).

As mentioned above, cross-docking and, in particular, model CDAP (3), is a difficult combinatorial optimization problem. Its treatment for problem solving is a young discipline, as a matter of fact, most of the literature has been published during the last 15 years. See comprehensive reviews in Agustina, Lee, and Pipiani (2010); VanBelle, Valckenaers, and Cattrysse (2012); Buijs, Vis, and Carlo (2014); Nassief (2017); Goodarzi et al. (2020). Cohen and Keren (2009) present one of the first models and a heuristic

for cross-docking. Zhu et al. (2009) introduce the binary quadratic model CDAP (3), whose notation and modeling have become a reference in the cross-docking literature. To the best of our knowledge, Guignard et al. (2012) present the first MILP mathematically equivalent model to CDAP (3); for that purpose, the Reformulation Linearization Technique (RLT1) introduced in Adams and Sherali (1986) has been used. The work outlines a local search heuristic that we have taken benefit from, and a convex hull-based metaheuristic designed explicitly for quadratic 0-1 problems with linear constraints. The testbed of instances that have been generated for validating the proposals has become another reference in the cross-docking literature. Nassief, Contreras, and As'ad (2016) introduce a modification of model CDAP by converting it in a mathematically equivalent binary linear one. For that purpose, a new structure is considered, where k is one of the, say, $|K|$ commodities to be sent from origin, say $m(k)$, to destination, say, $n(k)$. A new binary variable z_{kij} is introduced for the triplet (k, i, j), whose value 1 means commodity k enters the cross-dock through strip door i and exiting it through stack door j and otherwise, 0. The objective function (3a) and the constraints (1b) and (2b) are replaced in the new model with z-based ones, and redundant constraints based on (1a) and (2a) are appended for the triplet $(z_{kij}, x_{m(k),i}, y_{n(k),j})$. A Lagrangean Relaxation of those constraints results in a model to be decomposed in three types of independent submodels, one of them is further decomposed in $|K|$ trivially solved semi-assignment problems, and the other two are further decomposed in $|I| + |J|$ binary knapsack problems. The solution value at each Lagrangean iteration is a lower bound of the solution value of model CDAP. A lazy primal algorithm obtains a feasible solution from the Lagrangean one to be improved by a local search. The results are very good on the testbed in Guignard et al. (2012) as well as on new generated instances with larger dimensions. Nassief, Contreras, and Jaumard (2018) present two MILP models which are mathematically equivalent to model CDAP. One of such requires a column generation algorithm, so that new columns are appended to the associated so-named master problem relaxation. Guignard (2020) report some computational experiments on RTL1-based cross-docking. A very broad computational experiment have been reported in Gelareh et al. (2020) by considering alternatives to model CDAP, where different RLT1 models are compared whose main differences lie on the appending of a variety of tight redundant cuts.

Birim (2016); Enderer, Contardo, and Contreras (2017); Nassief (2017); Nasiri et al. (2018); Gunawan et al. (2021) integrate the cross-docking problem with the vehicle scheduling one from origin to destination such that the routes always begin and end at the doors of the cross-dock infrastructure. Simulated annealing and reduced variable neighborhood search algorithms, among others, are considered for problem solving. Yu, Jewpanya, and Kachitvichyanukul (2016) integrate cross-docks in a multiperiod product distribution network operations planning. Ye et al. (2018) present an approach for sequencing vehicles -in fact, trucks- to doors in order to minimize waiting time makespan; a particle swarm algorithm is considered jointly with a repair heuristic. Gaudioso, Monaco, and Sammarra (2021) deal also with the same type of sequencing and scheduling cross-docking problem; a Lagrangean decomposition scheme is considered jointly with two repair heuristics embedded in the Lagrangean process.

1.3. *Main contributions of the work and outline*

The solving approach for CDAP considered in this work takes benefit of the GAP structures (x, S)-(1) and (y, R)-(2) embedded in the original model. A summary of the main contributions of the work is as follows:

(i) Let us name model LIP to the MILP model that has been reported in Gelareh et al. (2020) as the most efficient mathematically equivalent one to CDAP (3), among of those many tested ones, in the testbed of instances in Guignard et al. (2012). An enlarged redundant model is introduced in our work, let us name it model LIP-e.

(ii) A split variable reformulation of model LIP-e is introduced to allow Lagrangean decomposition by dualizing the split variable constraints, in order to obtain lower bounds in reasonable computing time. That model is decomposed in two submodels, such that one is related to the strip doors, to be named strip submodel, and the other one is related to the stack doors, to be named stack submodel. A scheme for decoupling the strip and stack submodels is introduced by taking benefit of its Integer Linearized Property. The independent GAPs that result from each of those decoupling allow to provide strong lower bounds in very small computing time in large instances up to 25 nodes and 10 doors in each inbound and outbound side.

(iii) On the other hand, those two submodels are further tightened themselves by appending additional redundant cuts, to be named strip- and stack-related tightened submodels. It is proved that each of those two tightened submodels define, for zero Lagrange multipliers, two new formulations of the original model CDAP (3). It is remarkable that those submodels provide the solution in reasonable computing time for a subset of small and middle size instances taken from Guignard et al. (2012) (the largest ones have 12 nodes and 6 doors as defined above).

(iv) An improvement, so-named Local Search Heuristic (LSH), of the matheuristic algorithms LS1 and LS2 introduced in Guignard et al. (2012) for obtaining (hopefully, good) feasible solutions is provided.

The organization of the rest of the work is as follows: Section 2 deals with the models LIP and LIP-e, and the split variable reformulation for the latter one. Section 3 introduces the Lagrangean Decomposition of the split variable reformulation, the strip- and stack-related submodels and the further tightening ones on one hand and the decoupling scheme on the other one. Section 4 presents the Local Search Heuristic (LSH) for obtaining feasible solutions. Section 5 reports the results of a broad computational experiment that has been conducted for performing a benchmark among the schemes presented in the work by considering large-scale instances. And Section 6 draws the main conclusions and outlines a future research agenda. As a supplementary file, Appendix A details the proof of equivalence of the new formulations presented in Section 3; Appendix B details the results of the computational experimentation over a testbed of instances taken from Guignard et al. (2012) (instances with up to 15 nodes and 7 doors); Appendix C illustrates the procedure LSH by using a toy instance; and Appendix D gives the above equivalent formulations of CDAP by using that toy instance.

2. Equivalent Linearized Integer Programming models for CDAP model (3)

The binary quadratic model CDAP (3) can be transformed through the introduction of new variables into the mathematically equivalent one so-named Linearized mixed Integer Programming Problem (LIP). For that purpose, let us replace the quadratic term $x_{mi}y_{nj}$ with the continuous linear variable v_{minj}. The RTL1 formulation of the model satisfies its required property (i.e. $v_{minj} = 1$ if and only if $x_{mi} = y_{nj} = 1$), by appending the constraint system to be expressed as

$$x_{mi} + y_{nj} \leq 1 + v_{minj} \quad \forall m \in M, i \in I, n \in N, j \in J \tag{4a}$$

$$v_{minj} \leq x_{mi} \quad \forall m \in M, i \in I, n \in N, j \in J \tag{4b}$$

$$v_{minj} \leq y_{nj} \quad \forall m \in M, i \in I, n \in N, j \in J, \tag{4c}$$

from where it is easy to show that model LIP can be expressed

$$z^* = \min \sum_{m\in M} \sum_{i\in I} \sum_{n\in N} \sum_{j\in J} d_{ij} w_{mn}\, v_{minj} \tag{5a}$$

s. to

$$\sum_{j\in J} v_{minj} - x_{mi} = 0 \quad \forall m \in M, i \in I, n \in N \tag{5b}$$

$$\sum_{i\in I} v_{minj} - y_{nj} = 0 \quad \forall m \in M, n \in N, j \in J \tag{5c}$$

$$0 \leq v_{minj} \leq 1 \quad \forall m \in M,\ i \in I, n \in N, j \in J \tag{5d}$$

$$\text{constraint system GAP } (x, S)\ (1) \tag{5e}$$

$$\text{constraint system GAP } (y, R)\ (2). \tag{5f}$$

Note that (5b) (resp., (5c)) results from multiplying (2b) by x_{mi} (resp., (1b) by y_{nj}). Additionally, constraint (4a) is automatically satisfied for $x_{mi} = y_{nj} = 1$, where $m \in M, i \in I, n \in N, j \in J$.

Based on the computational results reported in Gelareh et al. (2020), model LIP (5) is the most efficient reformulation among the alternatives that have been experimented with by a straightforward using of a state-of-the-art solver in a broad testbed. That conclusion has also been confirmed by our own experimentation, see Section 5. However, we will also show that when the redundant constraints (6b) and (6c) are appended to model (5), the resulting enlarged reformulation, so-named LIP-e, expressed as

$$z^* = \min \sum_{m\in M} \sum_{i\in I} \sum_{n\in N} \sum_{j\in J} d_{ij} w_{mn}\, v_{minj} \tag{6a}$$

s. to

$$\sum_{n\in N} r_n v_{minj} - R_j x_{mi} \leq 0 \quad \forall m \in M, i \in I, j \in J \tag{6b}$$

$$\sum_{m\in M} s_m v_{minj} - S_i y_{nj} \leq 0 \quad \forall i \in I, n \in N, j \in J \tag{6c}$$

$$\text{constraint system (5b)} - \text{(5f)}, \tag{6d}$$

can drastically change the situation when the instances' dimension increases. The main reason is that it allows the model to be decomposed and decoupled in smaller submodels, see Section 3. Note: The constraints (6b) and (6c) result from multiplying (2a) and (1a) by x_{mi} and y_{nj}, resp.

2.1. *Split-variable reformulation of model LIP-e* (6)

Given the potential high dimensions of model LIP-e (6) in a real-life environment, it could be reformulated itself in order to consider a Lagrangean Decomposition (LD) scheme where the special structure of the problem is exploited for large-scale instances. For that purpose, a so-named split-variable reformulation may help to problem solving, since the approach may decompose the model in two smaller ones. A copy of the v-variables can be instrumental, then, let variable v'_{minj} be the copy of variable v_{minj}, so that a split-variable constraint should be appended to the model.

The split-variable reformulation of model LIP-e can be expressed

$$z^* = \min \quad \sum_{m\in M}\sum_{i\in I}\sum_{n\in N}\sum_{j\in J}\left(\frac{1}{2}d_{ij}w_{mn}\, v_{minj} + \frac{1}{2}d_{ij}w_{mn}\, v'_{minj}\right) \tag{7a}$$

s. to

$$v_{minj} - v'_{minj} = 0 \qquad \forall m \in M, i \in I, n \in N, j \in J \tag{7b}$$

$$\sum_{n\in N} r_n v_{minj} - R_j x_{mi} \leq 0 \qquad \forall m \in M, i \in I, j \in J \tag{7c}$$

$$\sum_{j\in J} v_{minj} - x_{mi} = 0 \qquad \forall m \in M, i \in I, n \in N \tag{7d}$$

$$\text{constraint system GAP } (x, S) \text{ (1)} \tag{7e}$$

$$0 \leq v_{minj} \leq 1 \qquad \forall m \in M, i \in I, n \in N, j \in J \tag{7f}$$

$$\sum_{m\in M} s_m v'_{minj} - S_i y_{nj} \leq 0 \qquad \forall i \in I, n \in N, j \in J \tag{7g}$$

$$\sum_{i\in I} v'_{minj} - y_{nj} = 0 \qquad \forall m \in M, n \in N, j \in J \tag{7h}$$

$$0 \leq v'_{minj} \leq 1 \qquad \forall m \in M, i \in I, n \in N, j \in J \tag{7i}$$

$$\text{constraint system GAP } (y, R) \text{ (2).} \tag{7j}$$

3. Decomposition based lower bounds to the solution value of reformulation (7)

This section presents some decomposition alternatives for providing (hopefully, strong) lower bounds of the optimal solution value of the split-variable reformulation (7). Subsection 3.1 introduces the strip- and stack-related LD submodels. Subsection 3.2 presents two new equivalent formulations of model LIP-e (5), so-named strip and stack tightened submodels, based on the tightening of the corresponding strip and stack-related ones, resp. Subsection 3.3 details the decoupling procedure of the strip- and stack-related submodels into smallers ones, by taking benefit of their Integer Linearization Property (ILP). Subsection

3.4 introduces a scheme for obtaining lower bounds based on the submodel decoupling.

3.1. *Lagrangean Decomposition (LD)*

Let us dualize the split-variable constraint (7b) in reformulation (7), for $m \in M$, $i \in I, n \in N$, $j \in J$, where $\delta_{minj} \forall m, i, n, j$ denote the unsigned Lagrangean multipliers vector of those constraints, be updated by considering the time-honored subgradient method. The resulting Lagrangean objective function can be expressed

$$z_{LD}(\delta) = \min \sum_{i \in I} \sum_{j \in J} \sum_{m \in M} \sum_{n \in N} [\frac{1}{2} d_{ij} w_{mn}\, v_{minj} + \frac{1}{2} d_{ij} w_{mn}\, v'_{minj} + \delta_{minj}(v_{minj} - v'_{minj})]. \quad (8)$$

Thus, by exploiting the x-related constraints on one side and the y-related constraints on the other one in (7c)-(7j), it results that the model can be decomposed in the submodels (9) and (10) to be expressed as follows:

Strip-related submodel with additional stack-related constraints

$$z_{LD}^{strip}(\delta) = \min \quad \sum_{m \in M} \sum_{i \in I} \sum_{n \in N} \sum_{j \in J} (\frac{1}{2} d_{ij} w_{mn} + \delta_{minj})\, v_{minj} \quad (9a)$$

s. to

$$\sum_{n \in N} r_n v_{minj} \leq R_j x_{mi} \quad \forall m \in M, i \in I, j \in J \quad (9b)$$

$$\sum_{j \in J} v_{minj} - x_{mi} = 0 \quad \forall m \in M, i \in I, n \in N \quad (9c)$$

$$v_{minj} \in \{0, 1\} \quad \forall m \in M, i \in I, n \in N, j \in J \quad (9d)$$

constraint system GAP (x, S) (1). (9e)

Stack-related submodel with additional strip-related constraints

$$z_{LD}^{stack}(\delta) = \min \quad \sum_{m \in M} \sum_{i \in I} \sum_{n \in N} \sum_{j \in J} (\frac{1}{2} d_{ij} w_{mn} - \delta_{minj})\, v'_{minj} \quad (10a)$$

s. to

$$\sum_{m \in M} s_m v'_{minj} \leq S_i y_{nj} \quad \forall i \in I, n \in N, j \in J \quad (10b)$$

$$\sum_{i \in I} v'_{minj} - y_{nj} = 0 \quad \forall m \in M, n \in N, j \in J \quad (10c)$$

$$v'_{minj} \in \{0, 1\} \quad \forall m \in M, i \in I, n \in N, j \in J \quad (10d)$$

constraint system GAP (y, R) (2). (10e)

Notice that the solution value $z_{LD}(\delta^{k+1})$ (8) of the δ-parametric model at a given Lagrangean iteration, say k (computed as the sum $z_{LD}^{strip}(\delta^{k+1}) + z_{LD}^{stack}(\delta^{k+1})$), is a lower bound of the solution value z^* of the

original model LIP (5). Let δ^k denote the Lagrange multipliers vector, $v(\delta^k)$ and $v'(\delta^k)$ be the optimal solution of the v- and v'-variables retrieved from the submodels (9) and (10), resp. Then, the subgradient, say s^k, can be expressed as the $|M| \times |I| \times |N| \times |J|$-dimensional vector $s^k = (v(\delta^k) - v'(\delta^k))$. So, by taking an upper bound of the solution value of (7), say $\tilde{z}$, and a positive factor $a^k \in (0, 2)$, see Geoffrion (1974), the $\delta-$multiplier vector updating can be expressed as

$$\delta^{k+1} = \delta^k + a^k \cdot \left(\frac{\tilde{z} - z_{LD}(\delta)}{||s^k||^2} \right) \cdot s^k.$$

Additionally, both submodels (9) and (10) can be tightened by appending the constraints (11a) and (11b), resp. Those new constraints are obtained by taking into account the information implied by constraints (7h) and (7d). The rationale behind it is as follows: It can be seen in constraints (7h) that the sum in the left-hand side (lhs) have the same value for all the indices $m, m' \in M : m \neq m'$, independently of the 0-1 value of variable y_{nj}, so, constraint (11a) is satisfied. In the same way, it can be seen in constraints (7d) that the lhs takes the same 0-1 value, independently of the 0-1 value of variable x_{mi}.

$$\sum_{i \in I} v_{minj} - \sum_{i \in I} v_{m'inj} = 0 \qquad \forall m, m' \in M : m \neq m', n \in N, j \in J \tag{11a}$$

$$\sum_{j \in J} v'_{minj} - \sum_{j \in J} v'_{min'j} = 0 \qquad \forall m \in M, i \in I, n, n' \in N : n \neq n'. \tag{11b}$$

The constraint system (11) is redundant in case it is appended to split-variable reformulation (7), without having any add-value. Moreover, its appending to the strip- and stack–related submodels (9) and (10) can have a tightening effect at the price of requiring more computing effort, see next.

3.2. *Strip- and stack-related tightened models. New equivalent formulations of model LIP* (5)

Let us consider the strip-related tightened submodel, for zero Lagrange multipliers, to be expressed as

$$z^*_{LD}(0) = \min \sum_{m \in M} \sum_{i \in I} \sum_{n \in N} \sum_{j \in J} d_{ij} w_{mn} \, v_{minj} \tag{12a}$$

s. to

constraint system (9b)-(9e) (12b)

constraint system (11a). (12c)

Proposition 3.1. *Model LIP* (5) *and strip- related tightened submodel* (12) *are equivalent in the following sense:*

(1) Given a feasible solution (x, y, v) *of model* (5)*, the corresponding solution* (x, v) *is feasible in model* (12) *with the same objective function value.*

(2) Conversely, given any feasible solution (x, v) *of model* (12)*, there exits a feasible solution* (x, y, v)

of model (5) *with the same objective function value.*

Proof. See Appendix A. □

A similar result of equivalence between formulations can be given in terms of the stack-related tightened submodel, for zero Lagrange multipliers, to be expressed as

$$z^*_{LD}(0) = \min \sum_{m\in M}\sum_{i\in I}\sum_{n\in N}\sum_{j\in J} d_{ij} w_{mn}\, v'_{minj} \tag{13a}$$

s. to

constraint system (10b)-(10e) (13b)

constraint system (11b). (13c)

Proposition 3.2. *Model LIP* (5) *and stack-related tightened submodel* (13) *are equivalent in the following sense:*

(1) Given a feasible solution (x, y, v) *of model* (5)*, the corresponding solution* (y, v) *is feasible in model* (13) *with the same objective function value.*
(2) Conversely, given any feasible solution (y, v) *of model* (13)*, there exits a feasible solution* (x, y, v) *of model* (5) *with the same objective function value.*

Proof. It is trivial by swapping the roles of M for N, I for J and x_{mi} for y_{nj} in the proof of Proposition 3.1. □

Corollary 3.3. *An alternative formulation of model LIP* (5) *is the one that results as the decomposition in the submodels* (9) *plus* (11a) *and* (10) *plus* (11b)*, with zero Lagrange multipliers, resp. In this case, the solution value coincides with* $z^*_{LD}(0)$.

Proof. It is followed from Propositions 3.1 and 3.2. □

Notice that the solution value of the split-variable reformulation (7) is also given by zero-Lagrangean solution value $z^*_{LD}(0)$.

3.3. *Using the ILP to decompose submodels* (9) *and* (10)

The ILP, see Geoffrion (1974); Guignard (2003), has proved to be very useful in the decoupling of subproblems into smaller ones. Note that the strip-related submodel (9) is in the v- and x-binary variables, so that for any pair (m, i) it happens that if $x_{mi} = 0$ then, constraints (9b) and (9c) force $v_{minj} = 0$ $\forall n \in N$, $j \in J$. On the other hand, if $x_{mi} = 1$, then those constraints define a feasible region merely dependent upon pair (m, i). So, the value of the objective function related to pair (m, i) takes the value 0 for $x_{mi} = 0$ and a value, say β_{mi}, when $x_{mi} = 1$.

Then, an x_{mi}-parametric subproblem for each pair (m, i) based on submodel (9) can be expressed as $\beta_{mi} = \min \sum_{n\in N} \sum_{j\in J} (\frac{1}{2} d_{ij} w_{mn} + \delta_{minj})\ v_{minj}$, subject to the corresponding constraints (9b) and (9c), which solution value is 0 for $x_{mi} = 0$. Thus, the optimal value of submodel (9) can be found by solving the following subproblem $z_{LD}^{strip}(\delta) = \min \sum_{m\in M} \sum_{i\in I} \beta_{mi} x_{mi}$, subject to the GAP constraints (x, S)-(1). Note that, after computing all the β_{mi} values, one per each pair (m, i), submodel (9) reduces to a GAP over the x-variables, which obviously is smaller in size and simpler in structure than the original one. Additionally, the subproblems to solve for obtaining the β-values are also GAPs, thus, they are as small and simple as the other one. A similar decomposition scheme can be applied to solve the stack-related submodel (10).

3.4. *On GAP-related sequential decoupling of the strip- and stack-related submodels-based lower bound*

By considering the decoupling scheme of submodels (9) and (10) as outlined above, this section presents the scheme for solving the related GAPs, at a given Lagrangean iteration.

Decoupling strip-related submodel (9)

Step 1

The solution value β_{mi} of the $N \times J$-binary variables in the (m, i)-GAP can be expressed

$$\beta_{mi} = \min \quad \sum_{n\in N} \sum_{j\in J} (\frac{1}{2} d_{ij} w_{mn} + \delta_{minj})\ v_{minj} \tag{14a}$$

s. to

$$\sum_{n\in N} r_n v_{minj} \le R_j \qquad \forall j \in J \tag{14b}$$

$$\sum_{j\in J} v_{minj} = 1 \qquad \forall n \in N \tag{14c}$$

$$v_{minj} \in \{0, 1\} \qquad \forall n \in N, j \in J. \tag{14d}$$

Step 2

Given the set of optimal values $\{\beta_{mi}\, \forall m \in M, i \in I\}$ retrieved from solving the models (14) in Step 1, the solution of submodel (9) can be equivalently obtained by the $M \times I$-binary variables-related GAP to be expressed as

$$z_{LD}^{strip}(\delta) = \min \sum_{m\in M} \sum_{i\in I} \beta_{mi} x_{mi} \tag{15a}$$

$$\text{s. to constraint system GAP } (x, S)\ (1). \tag{15b}$$

Thus, the solution of submodel (9) requires to solve $M \times I + 1$ GAPs.

Decoupling stack-related submodel (10)

A similar procedure can be carried out as for decoupling the strip-related submodel (9).

Step 1

The solution value α_{nj} of the $M \times I$-binary variables in the (n, j)-GAP can be expressed

$$\alpha_{nj} = \min \quad \sum_{m \in M} \sum_{i \in I} (\frac{1}{2} d_{ij} w_{mn} - \delta_{minj})\, v'_{minj} \tag{16a}$$

s. to

$$\sum_{m \in M} s_m v'_{minj} \leq S_i \qquad \forall i \in I \tag{16b}$$

$$\sum_{i \in I} v'_{minj} = 1 \qquad \forall m \in M \tag{16c}$$

$$v'_{minj} \in \{0, 1\} \qquad \forall m \in M, i \in I. \tag{16d}$$

Step 2

Given the set of optimal values $\{\alpha_{nj}\, \forall n \in N, j \in J\}$ retrieved from solving the models (16), the solution of submodel (10) can be equivalently obtained by the $N \times J$-binary variables-related GAP to be expressed as

$$z_{LD}^{stack}(\delta) = \min \sum_{n \in N} \sum_{j \in J} \alpha_{nj} y_{nj} \tag{17a}$$

$$\text{s. to constraint system GAP } (y, R) \text{ (2).} \tag{17b}$$

Thus, the solution of submodel (10) requires to solve $N \times J + 1$ GAPs.

As summary, and after executing the two-step scheme, a bound, say $\underline{z}$, of the optimal solution value z^* of model CDAP (3), obtained at a given Lagrangean iteration, can be expressed as $\underline{z} = z_{LD}^{strip}(\delta) + z_{LD}^{stack}(\delta)$. It is worth to point out that the above $M \times I + N \times J$ GAPs can be solved in embarrassing parallelization. For that purpose, let $\#th$ denote the number of threads in the available HW. Let also $\#g^{strip}$ and $\#g^{stack}$ denote the number of groups where, sequentially, the parallel execution of $\#th$ submodels (14) on one hand and (16) on the other one can be performed, resp., for each group. Those parameters can be computed as $\#g^{strip} = \lceil \frac{|M|.|I|}{\#th} \rceil$ and $\#g^{stack} = \lceil \frac{|N|.|J|}{\#th} \rceil$. So, a plausible estimation of the upper bound, say $t_{\underline{z}}$, of the overall parallel computing time can be expressed as

$$t_{\underline{z}} = \#it \cdot \big(\#g^{strip} \cdot \bar{t}_{mod-(14)} + \bar{t}_{mod-(15)} + \#g^{stack} \cdot \bar{t}_{mod-(16)} + \bar{t}_{mod-(17)}\big), \tag{18}$$

where $\#it$ is the number of Lagrangean iterations, $\bar{t}_{mod-(14)}$ and $\bar{t}_{mod-(16)}$ are the highest computing time required for solving a submodel in the GAPs (14) and the GAPs (16), resp., and $\bar{t}_{mod-(15)}$ and $\bar{t}_{mod-(17)}$ are the computing times required for solving the GAPs (15) and (17), resp.

4. The Local Search Heuristic (LSH) for obtaining feasible solutions

Given the difficulty of BQP, a matheuristic algorithm is considered in this section for obtaining feasible solutions for CDAP also taking benefit of the special structure of its two GAP constraint systems (x, S)-(1) and (y, R)-(2). The algorithm LSH is an improvement of the algorithm versions, so-named LS1 and LS2, presented in Guignard et al. (2012). It is proposed to split the original CDAP model (3) into two simpler submodels with linear objective functions, smaller dimensions and the same constraints and variables.

In particular, the quadratic objective function (3a) can be viewed either as a function of the y-variables or as a function of the x-ones, to be expressed as

$$z(x,y) = \sum_{m\in M}\sum_{i\in I}\left[\sum_{n\in N}\sum_{j\in J} d_{ij}w_{mn}y_{nj}\right]x_{mi} = \sum_{n\in N}\sum_{j\in J}\left[\sum_{m\in M}\sum_{i\in I} d_{ij}w_{mn}x_{mi}\right]y_{nj}. \tag{19}$$

Thus, for a given solution, say, $\hat{y} \equiv (\hat{y}_{nj}\, \forall n \in N,\, j \in J)$ of variables vector y, the solution value $z(x, \hat{y})$ can be retrieved from the model that can be expressed

$$z(x,\hat{y}) = \min \sum_{m\in M}\sum_{i\in I}\left[\sum_{n\in N}\sum_{j\in J} d_{ij}w_{mn}\hat{y}_{nj}\right]x_{mi} \tag{20}$$
s. to constraint system GAP (x, S) (1).

In a similar way, for a given solution, say, $\hat{x} \equiv (\hat{x}_{mi}\, \forall m \in M,\, i \in J)$ of variables vector x, the solution value $z(\hat{x}, y)$ can be retrieved from the model that can be expressed

$$z(\hat{x},y) = \min \sum_{n\in N}\sum_{j\in J}\left[\sum_{m\in M}\sum_{i\in I} d_{ij}w_{mn}\hat{x}_{mi}\right]y_{nj} \tag{21}$$
s. to constraint systems GAP (y, R) (2).

In case that the vectors $\hat{x}$ and $\hat{y}$ are feasible for their respective GAP models (20) and (21), then a feasible solution is obtained for CDAP model (3); its objective function value is $min\{z(x, \hat{y}), z(\hat{x}, y)\}$. Note that in an iterative procedure the value $\hat{y}$ of variables vector y can be generated *at random* at the first trial, but the x-solution of model (20) is still a feasible one, so that at the next trial the y-solution as retrieved from model (21) is also feasible. Continuing the procedure on that way, a local optimal solution of model (3) can be obtained through several trials until the solution value $z(x, \hat{y})$ cannot be improved any more, based on the initial solution $\hat{y}$. At each iteration, say k, a new initial solution $\hat{y}$ is generated *at random*. The procedure may continue until a given bound, say $\overline{k}$, on the number of iterations is reached. Let us notate the incumbent values of the variables vectors x and y of the procedure as $\overline{x}$ and $\overline{y}$, resp.,

such that the related cost can be expressed

$$\overline{z} = \sum_{m \in M} \sum_{i \in I} \sum_{n \in N} \sum_{j \in J} d_{ij} w_{mn} \overline{x}_{mi} \overline{y}_{nj}. \tag{22}$$

Note: The incumbent cost $\overline{z}$ can be useful as an upper bound in the B&C phase of the straightforward use of the state-of-the-art MILP optimizer for solving some models, see Section 5.

A similar procedure can be considered by taking $\hat{x}$ as the initial x-vector, being generated *at random*. It is worth to point out that the procedures composed of Steps 0-4 and Steps 5-9 of algorithm LSH can be executed in embarrassing parallelization. In this sense, let us recall that $\#th$ denotes de number of available threads. Let also $\#g$ denote the number of groups where, sequentially, the parallel execution of $\#th$ iterations of algorithm LSH is to be performed, by executing either Steps 0-4 or Steps 5-9. It can be expressed $\#g = \lceil \frac{2.\overline{k}}{\#th} \rceil$. So, a plausible estimation of the upper bound of the overall parallel computing time can be expressed as $\#g \cdot max\{\overline{t}_a, \overline{t}_b\}$, where $\overline{t}_a$ and $\overline{t}_b$ are the highest computing time required by any iteration of $a \equiv$Steps 0-4 and $b \equiv$Steps 5-9.

Algorithm 1 LSH. On obtaining an incumbent soln $(\overline{x}, \overline{y})$ with cost $\overline{z}$ (22) for model CDAP (3)

Step 0: (Initializing for $\hat{y}$-vector)
Set $k := 0$ and $\overline{z}_1 := \infty$.

Step 1: (Obtaining a potential incumbent soln. Start generating initial soln $\hat{y}$)
Generate *at random* a set of N integer numbers between 1 and J, say $\{r[n], n \in N\}$
Initialize $\hat{y}_{nr[n]} := 1$ and $\hat{y}_{nj} := 0$ for $j \neq r[n]$, for each $n \in N$.

Step 2: (Building and solving the GAP-strip related submodel (20)**, for soln vector $\hat{y}$)**
Keep the retrieved soln vector $\hat{x} := x$ and the soln value $z(x, \hat{y})$.

Step 3: (Building and solving the GAP-stack related submodel (21)**, for soln vector $\hat{x}$)**
Keep the retrieved soln vector $\hat{y} := y$ and soln value $z(\hat{x}, y)$.

Step 4: (Testing if $z(\hat{x}, y) < z(x, \hat{y})$)
If positive (i.e., potential incumbent soln improved) then goto Step 2.
If $z(\hat{x}, y) < \overline{z}_1 - \epsilon$ then update $\overline{z}_1 := z(\hat{x}, y)$, $\overline{x}_1 := \hat{x}$ and $\overline{y}_1 := \hat{y}$, where ϵ is a positive tolerance.
If $k < \overline{k}$ then update $k := k + 1$ and goto Step 1.

Step 5: (Initializing for $\hat{x}$-vector)
Reset $k := 0$ and set $\overline{z}_2 := \infty$.

Step 6: (Obtaining another potential incumbent soln. Start generating initial soln $\hat{x}$)
Generate *at random* a set of M integer numbers between 1 and I, say $\{r[m], m \in M\}$).
Initialize $\hat{x}_{mr[m]} := 1$ and $\hat{x}_{mi} := 0$ for $i \neq r[m]$, for each $m \in M$.

Step 7: (Building and solving the GAP-stack related submodel (21)**, for soln vector $\hat{x}$)**
Keep the retrieved vector $\hat{y} := y$ and soln value $z(\hat{x}, y)$.

Step 8 : (Building and solving the GAP-strip related submodel (20)**, for soln vector $\hat{y}$)**
Keep the retrieved soln vector $\hat{x} := x$ and soln value $z(x, \hat{y})$.

Step 9: (Testing if $z(x, \hat{y}) < z(\hat{x}, y)$)
If positive (i.e., potential incumbent soln improved) then goto Step 7.
If $z(x, \hat{y}) < \overline{z}_2 - \epsilon$ then update $\overline{z}_2 := z(x, \hat{y})$, $\overline{x}_2 := \hat{x}$ and $\overline{y}_2 := \hat{y}$.
If $k < \overline{k}$ then update $k := k + 1$ and goto Step 6.

Step 10: (Posting LSH incumbent soln)
If $\overline{z}_1 < \overline{z}_2$ then $\overline{z} := \overline{z}_1$, $\overline{x} := \overline{x}_1$, $\overline{y} := \overline{y}_1$; else $\overline{z} := \overline{z}_2$, $\overline{x} := \overline{x}_2$, $\overline{y} := \overline{y}_2$.

5. Computational experience

The experiments reported in this section were conducted on a Work Station Debian Linux (kernel v4.19.0), processors ES-2670 (20 threads), 2.50 Ghz and 128Gb RAM. The CPLEX v12.8 Concert Technology library has been used, embedded in a C/C++ experimental code, for solving the decomposition submodels as well as straightforward solving the full models.

For testing purposes a broad computational experiment has been performed by considering 55 instances in two testbeds. Testbed 1 is composed of a set of 35 benchmark instances in the literature, in fact proposed by Guignard et al. (2012). Testbed 2 is composed of a set of 20 much larger and difficult instances, where the first 15 ones are also taken from Guignard et al. (2012), and the other instances are generated for the experiment to validate the proposal in larger instances. The results obtained for the smaller ones, i.e., testbed 1, are presented in Appendix B. The rest of the section shows the results that have been obtained for testbed 2.

5.1. *LSH performance*

As it is shown in Appendix B, see Table B1, heuristic LSH provides the optimal solution for testbed 1 and non-difficult instances in testbed 2, improving on the results obtained in Guignard et al. (2012).

On the other hand, Table 1 shows the LSH's results for the difficult instances in testbed 2. The headings are as follows: xxxyySzz, instance code, where xx is the number of origin nodes and the number of destination nodes, yy is the number of strip doors and number of stack doors, and zz is the capacity slackness (in percentage); $\overline{z}$, incumbent solution value; $\overline{k}$, number of LSH iterations that are considered; $\#reruns$, the number of *reruns* of the algorithm; $\overline{\overline{z}}$, average of the incumbent solution values obtained in the reruns. $t_{\overline{z}}$ and $tt_{\overline{z}}$, estimation of the total computing time required by the reruns in case of parallelization and total computing time without parallelization, resp.; GR_{LSH}, goodness ratio of the LSH incumbent solution versus the one provided by straightforward CPLEX, expressed as $\frac{\overline{z}}{z^*_{CPX}}$, where z^*_{CPX} is the value provided by CPLEX, see also Table 3; and $LS1, LS2, CH$, incumbent solution values taken from Guignard et al. (2012) as obtained by the related heuristics.

Table 1. LSH incumbent solution value for difficult instances in testbed 2

Instance	$\overline{z}$	$\overline{k}$	$\#reruns$	$\overline{\overline{z}}$	$t_{\overline{z}}$	$tt_{\overline{z}}$	GR_{LSH}	$LS1$	$LS2$	CH
20x10S5	**29907**	25	40	30071.5	44.21	1748	0.999	29945	30033	29933
20x10S10	**29282**	20	40	29514.9	28.32	1087	0.992	29286	29286	29498
20x10S15	**29135**	25	40	29418.1	70.47	2767	0.997	29278	29184	29595
20x10S20	**28991**	20	10	29221.1	51.94	510	0.999	29130	28992	29089
20x10S30	**28533**	25	40	28733.8	73.35	2896	0.999	28800	28541	28571
25x10S5	44237	40	10	44382.5	389.21	3877	0.988			
25x10S10	43712	40	10	43946.5	234.14	2320	0.974			
25x10S15	43585	40	10	43692.5	283.30	2803	0.972			
25x10S20	43104	40	10	43283.0	235.79	2346	0.993			
25x10S30	42661	40	10	42771.7	263.20	2612	0.996			

It can be observed in the table that the ratio GR_{LSH} is always slightly smaller than 1, which means

that the solution quality is practically the same as the CPLEX one, even when considering the 2h time limit that has been allowed for the latter, see also Subsection 5.2.

Moreover, heuristic LSH provides a slighty smaller incumbent solution value $\overline{z}$ than any of the heuristics LS1, LS2 or CH for all the difficult instances in Guignard et al. (2012), see the values in bold.

Table 2 shows the main results for assessing the LSH's solution quality by considering the lower bound $\underline{z}$ retrieved from solving the strip- and stack-related submodels (9) and (10), resp., at the Lagrangean iteration where the multipliers updating have been stabilized. Hence, the new headings are as follows: $z_{LD}(0)$, lower bound provided by the strip- and stack-related submodels (9) and (10), with Lagrangean vector $\delta = 0$, by decoupling them into the submodels (14) and (15) for the former, and (16) and (17) for the latter; $t_{z_{LD}(0)}$ and $tt_{z_{LD}(0)}$, estimation of the required time (s), computed as (18) for $\#it = 1$ Lagrangean iteration in case of parallelization, and total computing time (s) without parallelization, resp., for obtaining lower bound $z_{LD}(0)$; $\underline{z}$, lower bound provided by the summation of the solution values of the submodels (15) and (17)(recall that it is tighter than $z_{LD}(0)$); $t_{\underline{z}}$ and $tt_{\underline{z}}$, estimation of the required time (s), computed as (18) in case of parallelization, and total computing time (s) without parallelization, resp., for obtaining lower bound $\underline{z}$; and $GAP_{LSH}\%$, optimality gap of the LSH incumbent solution $\overline{z}$, expressed as $100 \cdot \frac{\overline{z}-\underline{z}}{\overline{z}}$. It is worth to point out that $\underline{z}$ is obtained as the highest summation of the optimal solution values of the decoupled submodel (15) (that uses as data the solution of the submodels (14)) and the decoupled submodel (17) (that uses as data the solution of the submodels (16)), among the iterations.

Observe that even the total computing time (s) without parallelization for obtaining the LSH incumbent solution value $\overline{z}$ and the lower bound $z_{LD}(0)$, say, $tt_{\overline{z}} + tt_{z_{LD}(0)}$ is much smaller than the CPLEX time t^*_{CPX-5}, for the 10 largest instances, where $tt_{\overline{z}}$ and $tt_{z_{LD}(0)}$ are reported in Tables 1 and 2, resp.

Table 2. LSH solution quality for the instances in testbed 2

Instance	$\overline{z}$	$t_{\overline{z}}$	$z_{LD}(0)$	$t_{z_{LD}(0)}$	$tt_{z_{LD}(0)}$	$\underline{z}$	$t_{\underline{z}}$	$tt_{\underline{z}}$	$GAP_{LSH}\%$
15x6S5	13927	54.01	13140	5	18	13203	303	4609	6.29
15x6S10	13803	45.30	13066	3	9	13116	138	2434	6.23
15x6S15	13765	41.23	13057	2	8	13083	142	1958	5.93
15x6S20	13720	37.27	13044	1	10	13082	100	1159	5.53
15x6S30	13567	31.12	12950	1	5	12953	40	343	5.29
15x7S5	15054	31.20	13947	7	39	14107	449	6483	6.29
15x7S10	14810	28.32	13815	3	11	13888	216	5212	6.23
15x7S15	14657	38.02	13735	3	10	13788	186	3107	6.29
15x7S20	14514	43.14	13674	2	9	13712	103	1843	5.53
15x7S30	14409	40.53	13613	2	6	13647	106	1126	5.29
20x10S5	29907	44.21	26672	8	70	27018	368	6762	9.66
20x10S10	29282	28.32	26446	7	39	26573	489	4844	9.25
20x10S15	29135	70.47	26300	7	38	26437	423	3493	9.26
20x10S20	28991	51.94	26161	6	38	26276	403	4019	9.36
20x10S30	28533	73.35	25953	8	49	26035	524	4855	8.75
25x10S5	44237	389.21	39197	96	467	39233	1824	20042	11.31
25x10S10	43712	234.14	38977	14	105	39078	1001	7980	10.60
25x10S15	43585	283.30	38836	12	83	38942	999	15498	10.65
25x10S20	43104	235.79	38712	8	49	38801	736	10926	9.98
25x10S30	42661	263.20	38475	8	50	38540	678	7392	9.66

Additionally, we can also notice that the LD approach based on the split-variable reformulation (7)

strengths the zero-Lagrangean-based bound (i.e., $\underline{z}$ is higher than $z_{LD}(0)$). However, it is not worth the effort (represented in the alternative computing times $t_{\underline{z}}$ and $tt_{\underline{z}}$ versus $t_{z_{LD}(0)}$ and $tt_{z_{LD}(0)}$), given the tightness of $z_{LD}(0)$ with respect to the solution value z^* of the original model.

5.2. *Performance of straightforward CPLEX*

For the original binary quadratic model CDAP (3), the dimensions in the experiment by considering testbed 2 are from 180 up to 500 0-1 variables, from 180 to 500 constraints, and from 8100 up to 62500 quadratic terms.

Table 3 shows the main results when solving model LIP (5) by straightforward CPLEX for this set of instances. The headings are as follows: $\underline{z}_{CPX}$, lower bound obtained by CPLEX; z^*_{CPX}, and t^*_{CPX-5}, incumbent solution value and required computing time (s), resp.; z_{LP-5} and t_{LP-5}, related LP lower bound and required computing time (s); and symbol " means that the LP solution value as well as the related time have the same value for the set of instances with the same parameters, but the flow S-slackness percentage; $GAP_{CPX}\%$, optimality gap of the incumbent solution values z^*_{CPX}, expressed as $100 \cdot \frac{z^*_{CPX}-\underline{z}_{CPX}}{z^*_{CPX}}$; GR_{CPX}, goodness ratio of the CPLEX incumbent solution value z^*_{CPX} versus the LSH one $\overline{z}$, expressed as $\frac{z^*_{CPX}}{\overline{z}}$; and TR_{CPX}, time ratio, expressed as $\frac{t^*_{CPX-5}}{t_{\overline{z}}}$. Note that the higher $GR_{CPX} > 1$ and the higher $TR_{CPX} > 1$, the worse performance of CPLEX versus LSH.

Table 3. CPLEX performance for models LIP for the instances in testbed 2

Instance	$\underline{z}_{CPX}$	z^*_{CPX}	t^*_{CPX-5}	z_{LP-5}	t_{LP-5}	$GAP_{CPX}\%$	GR_{CPX}	TR_{CPX}
15x6S5	13927	13927	93.59	12544	0.38	0.00	1	1.73
15x6S10	13803	13803	89.74	"	"	0.00	1	1.98
15x6S15	13765	13765	48.70	"	"	0.00	1	1.18
15x6S20	13720	13720	14.20	"	"	0.00	1	0.38
15x6S30	13567	13567	13.43	"	"	0.00	1	0.43
15x7S5	15054	15054	121.77	12992	0.39	0.00	1	3.90
15x7S10	14810	14810	115.40	"	"	0.00	1	4.07
15x7S15	14657	14657	223.98	"	"	0.00	1	5.89
15x7S20	14514	14514	24.07	"	"	0.00	1	0.56
15x7S30	14409	14409	27.05	"	"	0.00	1	0.67
20x10S5	27692	29935	7200.00	24552	2.55	7.49	1.0009	162.86
20x10S10	27533	29521	7200.00	"	"	6.74	1.0082	254.24
20x10S15	27601	29208	7200.00	"	"	5.50	1.0025	102.17
20x10S20	26991	29020	7200.00	"	"	6.99	1.0010	138.62
20x10S30	27119	28566	7200.00	"	"	5.07	1.0012	98.16
25x10S5	38810	44766	7200.00	36824	6.59	13.31	1.0119	18.50
25x10S10	39035	44891	7200.00	"	"	13.05	1.0270	30.75
25x10S15	38963	44837	7200.00	"	"	13.10	1.0287	25.41
25x10S20	38411	43416	7200.00	"	"	11.53	1.0072	30.53
25x10S30	38194	42807	7200.00	"	"	10.78	1.0034	27.35

Observe in the table that CPLEX obtains the optimality in 50% of the instances (i.e., $GAP_{CPX}\% = 0$). On the other hand, given $GR_{CPX} = 1$, notice that LSH provides the same solution value with a smaller time (i.e., $TR > 1$). However, for the largest instances, GAP_{CPX} is not very small (nor it is GAP_{LSH}, see Table 2), and GR_{CPX} is close to 1, so, both approaches have practically the same solution quality. Moreover, TR is much higher than 1, i.e., the time required by CPLEX is much higher than the

one that does it by LSH.

6. Conclusions and future research agenda

Several schemes for obtaining lower bounds of the optimal solution value of CDAP, a difficult binary quadratic combinatorial problem, have been introduced. Those schemes take benefit of the GAP structures that are embedded in the model, in particular, in its seemingly most efficient linearized integer programming reformulation LIP (5).

Additionally, a heuristic algorithm has been provided for obtaining good feasible solutions in the two testbeds composed of 55 instances to consider for validating the proposals made in this work. It has been computationally proved that the overall scheme for providing the heuristic incumbent solution and strong lower bounds is very efficient. Those bounds are obtained from an efficient decomposition of model LIP-e (6), an enlarging of model LIP (5) that exploits its GAP structures. The proposed scheme obtains and proves the optimal solution in testbed 1, see Appendix B, composed of 35 instances in the experiment; those instances are a reference taken from the literature. Straightforward CPLEX also does the same for those instances, and requiring smaller computing time. The proposal obtains and proves an optimal solution in 10 out of the 20 instances included in testbed 2, the 5 largest ones are generated for testing it in larger instances. CPLEX also obtains and proves the optimal solution for those instances but requiring much higher computing time in most of them.

Neither the proposal, nor CPLEX can prove the optimality of the solution in the 10 largest instances in testbed 2; anyway, the optimality gaps are reasonable in both systems, although the proposal provides slightly better incumbent solution values, and it requires much less computing time than CPLEX.

Based on the computational experiment performed in this work, a provisional conclusion would be that the subgradient method-related LD lower bound of the split-variable reformulation (7) of the original model CDAP is stronger than the zero-Lagrangean one. However, it is not worth the effort, given the computing time required by the former and the tightness of the latter.

Finally, two new equivalent MILP models of CDAP are introduced. Both of them are obtained by tightening the corresponding strip- and stack-related submodels. It is also proved that any of these new formulations allow to obtain the solution value of the original model CDAP in reasonable computing time for the instances in testbed 1.

6.1. *Future research agenda*

The Simplicial Decomposition and Proximal Level methodology types would be investigated looking for even more efficient decomposition approaches for solving the original model CDAP. Another piece of research to work on is a further tightening of the decomposed submodels (9) and (10). The third line of research is related to the Cross-dock Door Design under uncertainty. A research on dynamic two-stage stochastic binary mixed quadratic optimization is planned, where the number and capacity of

the strip and stack doors is considered as first stage decisions. The uncertainty could lie on the sets of origin and destination nodes in a network and related commodity flow as well as on the door capacity diminishing due to disruptions along a time horizon; it would be represented in a finite set of scenarios. The assignment of nodes to doors for each scenario at a given time period as well as the commodity storage in the cross-docking infrastructure should be considered as second stage decisions along a time horizon in the scenarios.

Data Availability Statement

The data that support the findings of this study are available from the corresponding author, Escudero, L.F., upon reasonable request.

Acknowledgments

This research has been partially supported by the research projects RTI2018-094269-B-I00 and PID2021-122640OB-I00 (L.F. Escudero), and PID2019-104933GB-I00 from the Spanish Ministry of Science and Innovation, Grupo de Investigación EOPT (IT-1494-22) from the Basque Government, and GIU20/054 from University of the Basque Country (UPV/EHU) (M.A. Garín and A. Unzueta).

References

Adams, W.P., and Sherali, H.D. 1986. "A tight linearization and an algorithm for zero-one quadratic programming problems". *Management Science*, 32: 1274–1290.

Agustina, D., Lee, C.K.H., and Pipiani, R. 2010. "A review: Mathematical models for cross-docking planning". *International Journal of Production Research*, 2: 47–54.

Akkerman, F., Lalla-Ruiz, E., Mes, M., and Spitters, T. 2022. "Cross-docking: Current research versus industry practice and industry 4.0 adoption". *In Bondarouk, T. and Olivas-Luján, M.R. (Eds.), Smart Industry - Better Management*, 69–104, Emerald Group Holdings Co.

Billionnet, A., Elloumi, S., and Plateau, M.C. 2009. "Improving the performance of standard solvers via a tighter convex reformulation of constrained quadratic 0-1 programs". *Discrete Applied Mathematics*, 157: 1185–1197.

Birim, S. 2106. "Vehicle routing problem with cross docking: A simulated annealing approach". *Procedia – Social and Behavioral Sciences*, 235: 149–158.

Buijs, P., Vis, I.F.A., and Carlo, H.J. 2014. "Synchronization in cross-docking networks: A research classification and framework". *European Journal of Operational Research*, 239: 593—608.

Cesselli, A., Létocart, K., and Traversi, E. 2022. "Dantizg-Wolfe reformulation for binary quadratic problems". *Mathematical Programming Computation*, 14: 85–120.

Cohen, Y., and Keren, K. 2009. "Trailer to door assignment in a synchronous cross-dock operation". *International Journal of Logistics Systems and Management*, 5: 574–490.

Coindreau, M.-A., Gallay, O., Zufferey, N., and Laporte, G. 2021. "Inbound and outbound flow integration for cross-docking operations". *European Journal of Operational Research*, 294: 1153–1163.

Enderer, F., Contardo, C., and Contreras, I. (2017). "Integrating dock-door assignment and vehicle routing with cross-docking". *Computers and Operations Research*, 88: 30–43.

Fortet, R. (1960). "Application de l'algebre de boole en recherche operationelle". *Revue Francaise de Recherche Operationelle* 4: 17–26.

Gaudioso, M., Monaco, M.F., and Sammarra, M. (2021). "A Lagrangian heuristics for the truck scheduling problem in multi-door, multi-product Cross-Docking with constant processing time". *Omega*, 101: 102255.

Gelareh, S., Glover, F., Guemri, O., Hanafi, S., Nduwayo, P., and Todosijevic, R. 2020. "A comparative study of formulations for a cross-dock door assignment problem". *Omega*, 91: 102015. See instance testbeds in https://tinyurl.com/yb616v.

Geoffrion, A.M. 1974. "Lagrangean relaxation for integer programming". *Mathematical Programming Studies*, 2: 82–114.

Goodarzi, A.H., Zegordi, S.H., Alpan, G., Kamalabadi, I.N., and Kashan, A.H. 2020. "Reliable cross-docking location problem under the risk of disruptions". *Operational Research*, doi.org/10.1007/s12351-020-00583-5.

Guignard, M. 2003. "Lagrangean relaxation". *TOP*, 11: 151–228.

Guignard, M. 2020. "Strong RLT1 bounds from decomposable Lagrangean Relaxation for some quadratic 0–1 optimization problems with linear constraints". *Annals of Operations Research*, 286: 173–200.

Guignard, M., Hahn, P.M., Pessoa, A.A., and Cardoso da Silva, D. 2012. "Algorithms for the Cross-dock Door Assignment Problem". *In Proceedings of the Fourth International Workshop on Model-Based Metaheuristics 2012*. `www.optimization-online.org/DB-FILE/2012/06/3509.pdf`.

Gunawan, A., Widjaja, A.T., Vansteenwegen, P., and Yu, V.F. 2021. "A matheuristic algorithm for the vehicle routing problem with cross-docking". *Applied Soft Computing Journal*, 103: 107163.

Forger, G. 1995. "UPS starts world's premiere cross-docking operation". *Modern Material Handling*, 36: 36–38.

Kiani Mavi, R., Goh, M., Kiani Mavi, N., Jie, F., Brown, K., Biermann, S., and Khanfar, A. 2020. "Cross-Docking: a systematic literature review". *Sustainability*, 12, 4789; doi: 10.3390/su12114789.

Kinnear, E.K. 1997. "Is there any magic in cross-docking?" *Supply Chain Management*, 2: 49–52.

Ladier, A.L., and Alpan, G. 2006. "Cross-docking operations: Current research versus industry practice". *Omega*, 62: 145–162.

Napolitano, M. 2011. "Cross dock fuels growth at dots". *Logistics Management*, 50: 30–34.

Nasiri, M.M., Rahbari, A., Werner, F., and Karimi, R. 2018. "Incorporating supplier selection and order allocation into the vehicle routing and multi-cross-dock scheduling problem". *International Journal of Production Research*, 56: 6527–6552.

Nassief, W. 2017. *Cross-dock door assignments: Models, algorithm and extensions.* PhD Thesis, Concordia University. Montreal, Canada.

Nassief, W., Contreras, I., and As'ad, R. 2016. "A mixed-integer programming formulation and Lagrangean relaxation for the cross-dock door assignment problem". *International Journal of Production Research*, 454: 494–508.

Nassief, W., Contreras, I., and Jaumard, B. 2018. "A comparison of formulations and relaxations for cross-dock door assignment problems". *Computers & Operations Research*, 94: 76–88.

Ross, J. 1997. "Office Depot scores major cross-docking gains". *Sore Magazine*, 39.

Serrano, Ch., Delorme, X., and Dolgui A. 2021. "Cross-dock distribution and operationplanning for overseas delivery consolidation". *CIRP Journal of Manufacturing Science and Technology*, 33: 71–81.

Stalk, G., Evans, P., and Shulman, L.E. 1992. "Competing on capabilities: the new rules of corporate strategy". *Harvard Business Review*, 63.

VanBelle, J., Valckenaers, P., and Cattrysse, D. 2012. "Cross-docking: State-of-the-art". *Omega*, 40: 827–846.

Witt, C.E. 1998. "Crossdocking: Concepts demand choice". *Material Handling Engineering*, 53: 44–49.

Ye, Y., Li, J., Li, K., and Fu, H. 2018. "Cross-docking truck scheduling with product unloading/loading constraints based on an improved particle swarm optimisation algorithm". *International Journal of Production Research*, 56: 5365–5385.

Yu, V.F., Jewpanya, P., and Kachitvichyanukul, V. 2016. "Particle swarm optimization for the multi-period cross-docking distribution problem with time windows". *International Journal of Production Research*, 54: 509–525.

Zhu, Y.R., Hahn, P.M., Liu, Y., and Guignard, M. 2009. "New approach for the cross-dock door assignment problem". *In Proceedings of the XLI Brazilian Symposium on Operations Research.* Porto Seguro, Bahia, Brazil.

Appendix A. Proof of Proposition 3.1

Proof. (1) For a given feasible solution (x, y, v) of model LIP (5), it is trivial to show that (x, v) is a feasible solution of model (12) and the value of their objective functions match as long as the constraints (5b) and (5e) are part of (12), while constraints (5c) and (5f) imply the satisfaction of (12c) and (9b), resp.

(2) Conversely, to show the other direction of the equivalence, given that model (5) contains the constraints of model (12) with the (x, v)-variables, we have to show that from the satisfaction of constraints (9b) and (12c) by a given solution (x, v) of (12), it can be obtained a feasible solution (x, y, v), with the y-variables satisfying (5c) and (5f), which guarantees that the objective functions (5) and (12) have the same value.

On the one hand, given that the x-variables satisfy constraints (1b) in (9e), it follows that

$$\forall m \in M \; \exists! \; i = i(m) : x_{mi(m)} = 1 \;\; \wedge \;\; x_{mi} = 0 \; \forall i \neq i(m). \tag{A1}$$

And, on the other hand, from the knapsack constraints (1a) in (9e), it follows that for each $i \in I$ there exist, say, $K(i)$ subsets of indices $M_1, \cdots, M_{K(i)} \subseteq M$, such that for each $k \in \{1, \cdots, K(i)\}$, the variables $x_{mi(m)}$ can simultaneously take the value 1 $\forall m \in M_{k(i)}$. Then, taking into account (A1) and the feasibility of constraint system (9e), it can be guaranteed that there exist a number of partitions $K \geq 1$ with $K \times |I|$ subsets $M'_1, \cdots, M'_{K \times |I|} \subseteq M$, such that any feasible x-solution satisfies

$$\begin{aligned} x_{m'i(m')} = 1 \, \forall m' \in M'_k, \qquad x_{mi(m')} = 0, \;\; \forall m \in M - M'_k \\ x_{m'i} = 0 \, \forall i \in I, i \neq i(m'), \forall m' \in M'_k \end{aligned} \tag{A2}$$

for a given subset M'_k, $k \in K \times |I|$.

Now, and for each $m \in M$, let us consider the pair $(m, i(m))$, such that $x_{mi(m)} = 1$. In this case, from the set of constraints (9c), it follows that

$$\forall n \in N \; \exists! \; j = j(n) : v_{mi(m)nj(n)} = 1 \;\; \wedge \;\; v_{mi(m)nj} = 0 \; \forall j \neq j(n). \tag{A3}$$

And, also, for each pair (m, i), with $i \neq i(m)$ such that $x_{mi} = 0$, then $v_{minj} = 0 \; \forall n \in N, j \in J$. In short, we have the following implications,

$\forall m \in M$

$$\begin{aligned} \exists!\, i = i(m) : x_{mi(m)} = 1 \quad &\Rightarrow \quad \forall n \in N\ \exists!\, j = j(n) : v_{mi(m)nj(n)} = 1 \\ &\qquad \wedge\ v_{mi(m)nj} = 0, j \neq j(n) \\ \forall i \neq i(m) : x_{mi} = 0 \quad &\Rightarrow \quad v_{minj} = 0\, \forall n \in N, \forall j \in J \end{aligned} \tag{A4}$$

Moreover, for each pair $(m, i(m))$, such that $x_{mi(m)} = 1$ and from constraint (9b), for each $j \in J$ there exist, say, $L(j)$ subsets of indices $N_1, \cdots, N_{L(j)} \subseteq N$, such that the variables $v_{mi(m)nj(n)}$ can simultaneously take the value 1 $\forall n \in N_{l(j)}$. Taking into account (A4) and the feasibility of constraint system (9b)-(9c), it is follows the existence of $L \geq 1$ partitions with $L \times |J|$ subsets $N'_1, \cdots, N'_{L\times|J|} \subseteq N$, such that the set of feasible values for the v-solution satisfies

$$\begin{aligned} v_{mi(m)n'j(n')} = 1\ \ \forall n' \in N'_l \qquad v_{mi(m)nj(n')} = 0\ \ \forall n \in N - N'_l \\ v_{mi(m)n'j} = 0\ \ \forall j \in J, j \neq j(n'), \forall n' \in N'_l, \end{aligned} \tag{A5}$$

for a given subset N'_l, $l \in L \times |J|$. Then, we can see what happens when $\sum_{i\in I} v_{minj}$ in (12c) has the same value for all m. Given that the (x, v)-variables satisfy (9b), then by adding these constraints for $i \in I$ and taking into account that $\sum_{i\in I} v_{minj}$ has the same value independently of m, it results that the following relationship is satisfied

$$\sum_{n\in N} r_n(\sum_{i\in I} v_{minj}) \leq R_j(\sum_{i\in I} x_{mi}) = R_j \qquad \forall j \in J, \tag{A6}$$

and we can write $\sum_{i\in I} v_{minj} = v_{\bullet\bullet nj} = y_{nj}\ \forall n, j$. Notice that $y_{nj} \in \{0, 1\}$ is feasible for constraints (5c) and (5f) of model LIP (5) and, additionally, it is the optimal solution when the corresponding v-solution does it.

□

Appendix B. Computational results for testbed 1 and non-difficult instances in testbed 2

The dimensions of instances in testbed 1 are from $|M| = |N| = 8$ and $|I| = |J| = 4$ up to 12 and 6, resp. The 25% of the elements of the square flow matrix w_{mn}, for $m \in M$, $n \in N$, are generated *at random* integer (positive) values in the range from 10 to 50. The elements in the unit distance matrix ($d_{ij}\, \forall i \in I,\ j \in J$) range from 8 to $8 + |I| - 1$, where $d_{ij} = d_{ji}$ represents a symmetric pattern for the distance between strip door i and stack door j. The capacity of each door is set to the total flow coming from all origins/destinations divided by the total number of strip/stack related doors, plus a percentage, say, $S\%$ of the slackness of the total flow, where $S \in \{5, 10, 15, 20, 30\}$.

Table B1 shows the results of heuristic LSH, presented in Section 4, for testbed 1 and non-difficult instances in testbed 2.

Table B1. LSH performance: Optimal soln in testbed 1 and non-difficult instances in testbed 2

Instance	$\overline{z}$	$\overline{k}$	$\#reruns$	$\overline{\overline{z}}$	$t_{\overline{z}}$	$tt_{\overline{z}}$	$LS1$	$LS2$	CH
8x4S5	5174	5	10	5174.8	1.42	14	5174	5174	5174
8x4S10	5169	5	10	5169.0	1.65	16	5169	5169	5169
8x4S15	5112	5	10	5117.0	1.21	12	5112	5112	5112
8x4S20	5086	5	10	5108.1	1.02	10	5086	5086	5086
8x4S30	5063	5	10	5077.0	1.35	13	5063	5063	5063
9x4S5	6047	5	10	6050.0	2.53	25	6047	6047	6047
9x4S10	6027	5	10	6039.0	1.97	19	6027	6027	6027
9x4S15	5976	5	10	5985.4	1.24	12	5976	5976	5976
9x4S20	5937	5	10	5974.0	1.35	13	5937	5937	5937
9x4S30	5904	8	10	5907.2	2.24	22	5904	5904	5904
10x4S5	6518	5	10	6530.3	2.84	26	6518	6518	6518
10x4S10	6325	5	10	6339.0	1.49	14	6325	6325	6325
10x4S15	6296	5	10	6310.9	1.37	13	6296	6296	6296
10x4S20	6267	5	10	6292.4	1.26	12	6267	6267	6267
10x4S30	6193	8	10	6213.3	1.68	16	6193	6193	6193
10x5S5	6616	5	10	6623.4	3.17	31	6166	6166	6166
10x5S10	6476	20	2	6476.0	6.59	13	6476	6476	6476
10x5S15	6397	5	10	6426.5	2.12	20	6397	6397	6397
10x5S20	**6342**	10	10	6376.2	3.13	30	6374	6363	6354
10x5S30	**6308**	30	2	6316.0	8.57	17	6324	6333	6333
11x5S5	7812	5	5	7822.0	4.36	21	7812	7812	7812
11x5S10	7572	5	5	7608.4	4.72	20	7572	7572	7572
11x5S15	**7535**	10	5	7550.8	5.27	26	7542	7542	7542
11x5S20	**7439**	10	5	7483.0	3.93	18	7465	7465	7478
11x5S30	7420	10	5	7455.0	5.23	25	7428	7424	7420
12x5S5	8072	15	3	8077.0	13.21	39	8072	8072	8072
12x5S10	7978	20	5	7994.2	8.14	36	7978	7988	7991
12x5S15	7939	20	10	7964.9	9.21	89	7939	7939	7939
12x5S20	7939	20	10	7958.7	9.97	91	7939	7939	7939
12x5S30	**7923**	20	10	7938.1	8.94	81	7942	7925	7925
12x6S5	10891	20	20	10900.8	16.51	302	10891	10894	10894
12x6S10	**10456**	20	20	10501.3	12.12	232	10475	10496	10480
12x6S15	**10362**	40	20	10400.4	21.01	400	10395	10420	10374
12x6S20	**10312**	20	20	10365.8	9.31	168	10331	10339	10323
12x6S30	10228	20	20	10315.0	9.27	168	10228	10228	10277
15x6S5	13927	40	40	13968.0	54.01	2155	13960	13927	13983
15x6S10	**13803**	40	40	13853.8	45.30	1796	13856	13852	13872
15x6S15	13765	40	40	13816.1	41.23	1636	13769	13799	13765
15x6S20	**13720**	40	40	13765.6	37.27	1460	13769	13754	13720
15x6S30	13567	40	40	13622.6	31.12	1224	13567	13567	13626
15x7S5	15054	25	20	15080.1	31.20	600	15054	15054	15096
15x7S10	14810	25	40	14841.6	28.32	1098	14841	14818	14810
15x7S15	**14657**	40	20	14693.0	38.02	748	14678	14680	14678
15x7S20	**14514**	40	20	14566.1	43.14	846	14596	14533	14533
15x7S30	**14409**	40	20	14432.8	40.53	798	14415	14423	14414

The headings are as follows: $\overline{z}$, incumbent solution value; $\overline{k}$, number of LSH iterations that are considered; $\#reruns$, the number of *reruns* of the algorithm; $\overline{\overline{z}}$, average of the incumbent solution values obtained in the reruns; $t_{\overline{z}}$ and $tt_{\overline{z}}$, estimation of the total computing time required by the reruns in case of parallelization and total computing time without parallelization, resp.; and $LS1, LS2, CH$, incumbent solution values taken from Guignard et al. (2012) as obtained by the related heuristics.

The LSH incumbent solution values shown in Table B1 result in $GR_{LSH} = 1$ and, since the CPLEX

solution is optimal for those instances, LSH one is also optimal. It is worth to point out that LSH outperforms the heuristics LS1, LS2 and CH in 13 out of the 45 instances, see the values in bold.

Table B2 provides the main results of the performance of heuristic LSH for the instances in testbed 1 in comparison with the lower bounds of the optimal solution of the original model LIP (5).

Table B2. Lower bound-based guarantee of LSH soln optimality for testbed 1

Instance	$\overline{z}$	$t_{\overline{z}}$	$z_{LD}(0)$	$z^*_{LD}(0)$	$t^*_{LD}(0)$
8x4S5	5174	1.42	5083	5174	1.05
8x4S10	5169	1.65	5055	5169	0.98
8x4S15	5112	1.21	5055	5112	0.79
8x4S20	5086	1.02	5052	5086	0.85
8x4S30	5063	1.35	4995	5063	0.91
9x4S5	6047	2.53	5883	6047	1.30
9x4S10	6027	1.97	5873	6027	1.55
9x4S15	5976	1.24	5844	5809	1.99
9x4S20	5937	1.35	5809	5937	1.60
9x4S30	5904	2.24	5789	5904	1.52
10x4S5	6518	2.84	6235	6518	2.54
10x4S10	6325	1.49	6153	6325	2.49
10x4S15	6296	1.37	6141	6296	2.11
10x4S20	6267	1.26	6136	6267	2.59
10x4S30	6139	1.68	6107	6139	2.63
10x5S5	6616	3.17	6273	6616	21.06
10x5S10	6476	6.59	6228	6476	9.74
10x5S15	6397	2.12	6204	6396	14.53
10x5S20	6342	3.13	6194	6342	5.28
10x5S30	6308	8.57	6161	6308	7.13
11x5S5	7812	4.36	7376	7818	47.45
11x5S10	7572	4.72	7245	7572	13.68
11x5S15	7535	5.27	7241	7535	17.22
11x5S20	7439	3.93	7219	7439	12.70
11x5S30	7420	5.23	7188	7420	15.69
12x5S5	8072	13.21	7719	8072	43.90
12x5S10	7978	8.14	7637	7978	20.35
12x5S15	7939	9.21	7632	7939	27.51
12x5S20	7939	9.97	7643	7939	32.15
12x5S30	7923	8.94	7616	7923	39.33
12x6S5	10891	16.51	7719	10891	252.32
12x6S10	10456	12.12	7637	10456	81.75
12x6S15	10362	21.01	7632	10362	75.48
12x6S20	10312	9.31	7643	10312	76.52
12x6S30	10228	9.27	7616	10228	68.48

The new additional headings are as follows: $z_{LD}(0)$, lower bound provided by the strip- and stack-related submodels (9) and (10), with Lagrangean vector $\delta = 0$, by decoupling them into the submodels (14) and (15) for the former, and (16) and (17) for the latter; $z^*_{LD}(0)$, optimal solution value of any of the equivalent submodels (12) or (13) (recall that the solution value of each of those submodels coincides with the solution value of model LIP (5), see Propositions 3.1 and 3.2 in Section 3.2); and $t^*_{LD}(0)$, required computing time (s) to obtain $z^*_{LD}(0)$. It can be observed that $\overline{z} = z^*_{LD}(0)$ and, so, the optimality gap, say $GAP_{LHS}\%$, of the LSH incumbent solution, expressed as $100 \cdot \frac{\overline{z} - z^*_{LD}(0)}{\overline{z}}$, is zero (i.e., the optimality of the LSH solution is guaranteed for all of those instances), $z_{LD}(0) < \overline{z}$ and, in any case, $z_{LD}(0) < z^*_{LD}(0)$.

On the other hand, and for the original binary quadratic model CDAP (3), the dimensions in testbed 1 are from 64 up to 144 0-1 variables, from 64 to 144 constraints, and from 1024 up to 5184 quadratic terms. Table B3 shows the main results that are obtained when solving the models LIP (5) and LIP-e (6) by straightforward CPLEX for the instances in testbed 1. The headings are as follows: z^*_{CPX}, and t^*_{CPX-5} and t^*_{CPX-6}, incumbent solution value and required computing time (s), resp.; $z_{LP-\bullet}$ and $t_{LP-\bullet}$, for $\bullet = \{5, 6\}$, related LP lower bound and required computing time (s); and symbol " means that the LP solution value as well as the related time have the same value for the set of instances with the same parameters, but the flow S-slackness percentage. Observe that CPLEX obtains the optimal solution in both models LIP (5) and LIP-e (6) for all instances in testbed 1 (as heuristic LSH does it, see Table B1, as well as the decomposition approach for obtaining the lower bound $z^*_{LD}(0)$ of model (6), see Table B2). Moreover, it is not a surprise the small time required to solve model LIP (5) in comparison with model LIP-e (6). However, surprisingly, the integrality relaxation of the variables allows to have the same solution value and time for the high variability on the S-slackness of the total flow, varying from 5 to 50%. In any case, notice that $z_{LP-6} > z_{LP-5}$ in the 35 instances.

Table B3. CPLEX performance. Optimal soln for models LIP in testbed 1

Instance	z^*_{CPX}	t^*_{CPX-5}	z_{LP-5}	t_{LP-5}	t^*_{CPX-6}	z_{LP-6}	t_{LP-6}
8x4S5	5174	0.68	4824	0.02	2.87	5009	0.19
8x4S10	5169	0.90	"	"	3.14	4998	0.17
8x4S15	5112	0.53	"	"	1.74	4986	0.17
8x4S20	5086	0.77	"	"	1.64	4975	0.14
8x4S30	5063	0.94	"	"	2.16	4955	0.18
9x4S5	6047	1.16	5584	0.02	5.56	5828	0.20
9x4S10	6027	0.99	"	"	4.39	5808	0.18
9x4S15	5976	1.00	"	"	3.49	5789	0.20
9x4S20	5937	0.71	"	"	2.19	5575	0.18
9x4S30	5904	1.40	"	"	3.11	5749	1.19
10x4S5	6518	1.16	5960	0.05	8.87	6110	0.23
10x4S10	6325	1.32	"	"	8.38	6093	0.24
10x4S15	6296	1.41	"	"	8.26	6079	0.22
10x4S20	6267	1.07	"	"	4.44	6067	0.21
10x4S30	6193	1.15	"	"	4.56	6048	0.20
10x5S5	6616	2.53	5968	0.07	34.56	6191	0.39
10x5S10	6476	2.82	"	"	31.26	6170	0.37
10x5S15	6397	2.21	"	"	31.52	6153	0.37
10x5S20	6342	2.54	"	"	25.00	6134	0.37
10x5S30	6308	1.67	"	"	20.71	6108	0.33
11x5S5	7812	2.40	6968	0.13	53.95	7244	0.47
11x5S10	7572	2.19	"	"	38.54	7213	0.49
11x5S15	7535	1.91	"	"	36.04	7190	0.48
11x5S20	7439	2.19	"	"	29.34	7171	0.45
11x5S30	7420	1.85	"	"	46.35	7141	0.47
12x5S5	8072	3.09	7408	0.13	89.79	7648	0.73
12x5S10	7978	2.45	"	"	79.07	7623	0.59
12x5S15	7939	2.75	"	"	78.43	7605	0.59
12x5S20	7939	3.46	"	"	71.69	7586	0.60
12x5S30	7923	2.51	"	"	129.77	7558	0.54
12x6S5	10891	7.18	9472	0.18	636.65	9859	1.06
12x6S10	10456	4.83	"	"	272.03	9825	0.99
12x6S15	10362	6.23	"	"	341.12	9801	0.96
12x6S20	10312	6.28	"	"	266.18	9778	0.96
12x6S30	10228	5.32	"	"	214.26	9736	0.98

Appendix C. Illustrative usage of procedure LSH in a toy instance

Let us consider the following CDAP toy instance, with $|M| = |N| = 4$ and $|I| = |J| = 2$, where the flow matrix (w_{mn}) is given by $\begin{pmatrix} 0 & 0 & 26 & 0 \\ 22 & 0 & 0 & 0 \\ 26 & 32 & 0 & 50 \\ 0 & 47 & 0 & 31 \end{pmatrix}$, the distance matrix (d_{ij}) is equal to $\begin{pmatrix} 8 & 9 \\ 9 & 8 \end{pmatrix}$, being the corresponding vectors for the strip and stack door capacities, such that $S_i = 129\,\forall i \in I$ and $R_j = 129 \forall j \in J$, resp., both of them computed using a slack of 10%. So, the corresponding model CDAP (3) can be expressed

$$
\begin{aligned}
&\min \sum_{m\in M}\sum_{i\in I}\sum_{n\in N}\sum_{j\in J} d_{ij}w_{mn}x_{mi}y_{nj} \\
&\text{s. to} \\
&26x_{1i} + 22x_{2i} + 108x_{3i} + 78x_{4i} \leq 129 \quad \forall i \in \{1,2\} \\
&x_{m1} + x_{m2} = 1 \quad \forall m \in \{1,\cdots,4\} \\
&48y_{1j} + 79y_{2j} + 26y_{3j} + 81y_{4j} \leq 129 \quad \forall j \in \{1,2\} \qquad \text{(C1)} \\
&y_{n1} + y_{n2} = 1 \quad \forall n \in \{1,\cdots,4\} \\
&x_{mi} \in \{0,1\} \quad \forall m \in \{1,\cdots,4\},\, i \in \{1,2\} \\
&y_{nj} \in \{0,1\} \quad \forall n \in \{1,\cdots,4\},\, j \in \{1,2\}.
\end{aligned}
$$

Let us consider $\overline{k} = 2$ iterations and $\#reruns = 1$ in heuristic LSH, see Section 4; the best upper bound obtained is $\overline{z} = 1957$, which coincides with the optimal solution value of the original model (C1). Also, a multiple solution has been identified. One of these solutions, which appears twice, at iteration $k = 1$ of Steps 1-4 and Steps 6-9, see Table C1, is given by

$\overline{x}_{11} = \overline{x}_{21} = \overline{x}_{32} = \overline{x}_{41} = 1$, and, otherwise, $\overline{x}_{mi} = 0$;

$\overline{y}_{12} = \overline{y}_{21} = \overline{y}_{31} = \overline{y}_{42} = 1$, and otherwise, $\overline{y}_{nj} = 0$.

Another one, which appears at iteration $k = 2$ of Steps 1-4, is given by

$\overline{x}_{12} = \overline{x}_{22} = \overline{x}_{31} = \overline{x}_{41} = 1$, and, otherwise, $\overline{x}_{mi} = 0$;

$\overline{y}_{11} = \overline{y}_{22} = \overline{y}_{32} = \overline{y}_{41} = 1$, and otherwise, $\overline{y}_{nj} = 0$.

Appendix D. Equivalent formulations of CDAP in a toy instance

In this section it is shown how the solution of model CDAP (C1) can be obtained by solving a strip-related enlarged GAP submodel (9) with the same objective function as in model (C1) plus constraints (11a) (i.e., the equivalent formulation based on the strip-related enlarged submodel), see model (D1). A similar proof can be done, considering the stack-related enlarged GAP submodel (10) with the same

Table C1. Performance of the procedure LSH on a toy instance

- **Step 0**: $k := 0$, $\overline{z}_1 := \infty$.
- **Step 1**: $\hat{y}_{12} = \hat{y}_{21} = \hat{y}_{32} = \hat{y}_{42} = 1$; otherwise, $\hat{y}_{nj} = 0$.
- **Step 2**: $z(x, \hat{y})$ =1983. $\hat{x}_{11} = \hat{x}_{21} = \hat{x}_{32} = \hat{x}_{41} = 1$; otherwise, $\hat{x}_{mi} = 0$.
- **Step 3**: $z(\hat{x}, y)$ =1957. $\hat{y}_{12} = \hat{y}_{21} = \hat{y}_{31} = \hat{y}_{42} = 1$; otherwise, $\hat{y}_{nj} = 0$.
- **Step 4**: $z(\hat{x}, y) =$ **1957**<1983 $= z(x, \hat{y})$. $\overline{z}_1 =$**1957**. $\overline{x}_1 = \hat{x}$, $\overline{y}_1 = \hat{y}$, $k := 1 < \overline{k}$, goto Step 1.
- **Step 1**: $\hat{y}_{11} = \hat{y}_{21} = \hat{y}_{32} = \hat{y}_{42} = 1$; otherwise, $\hat{y}_{nj} = 0$.
- **Step 2**: $z(x, \hat{y}) = 1972$. $\hat{x}_{12} = \hat{x}_{22} = \hat{x}_{31} = \hat{x}_{42} = 1$; otherwise, $\hat{x}_{mi} = 0$.
- **Step 3**: $z(\hat{x}, y) = 1957$. $\hat{y}_{11} = \hat{y}_{22} = \hat{y}_{32} = \hat{y}_{41} = 1$; otherwise, $\hat{y}_{nj} = 0$.
- **Step 4**: $z(\hat{x}, y) =$ **1957**<1972 $= z(x, \hat{y})$, $k := 2 = \overline{k}$.
- **Step 5**: $k := 0$, $\overline{z}_2 := \infty$.
- **Step 6**: $\hat{x}_{11} = \hat{x}_{22} = \hat{x}_{32} = \hat{x}_{41} = 1$; otherwise, $\hat{x}_{mi} = 0$.
- **Step 7**: $z(\hat{x}, y) = 1935$. $\hat{y}_{12} = \hat{y}_{21} = \hat{y}_{31} = \hat{y}_{42} = 1$; otherwise, $\hat{y}_{nj} = 0$.
- **Step 8**: $z(x, \hat{y}) = 1957$. $\hat{x}_{11} = \hat{x}_{21} = \hat{x}_{32} = \hat{x}_{41} = 1$; otherwise, $\hat{x}_{mi} = 0$.
- **Step 9**: $z(x, \hat{y}) =$ **1957**>1935 $= z(\hat{x}, y)$. $\overline{z}_2 =$**1957**. $\overline{x}_2 = \hat{x}$, $\overline{y}_2 = \hat{y}$, $k := 1 < \overline{k}$, goto Step 6.
- **Step 6**: $\hat{x}_{11} = \hat{x}_{22} = \hat{x}_{31} = \hat{x}_{41} = 1$; otherwise, $\hat{x}_{mi} = 0$.
- **Step 7**: $z(\hat{x}, y) = 1977$. $\hat{y}_{12} = \hat{y}_{22} = \hat{y}_{31} = \hat{y}_{41} = 1$; otherwise, $\hat{y}_{nj} = 0$.
- **Step 8**: $z(x, \hat{y}) = 1987$. $\hat{x}_{12} = \hat{x}_{22} = \hat{x}_{31} = \hat{x}_{42} = 1$; otherwise, $\hat{x}_{mi} = 0$.
- **Step 9**: $z(x, \hat{y}) =$ **1987**>1977 $= z(\hat{x}, y)$, $k := 2 = \overline{k}$.
- **Step 10**: $\overline{z} = \overline{z}_1 = \overline{z}_2 =$**1957**. $\overline{x}^1 = \overline{x}_1$, $\overline{y}^1 = \overline{y}_1$(and, $\overline{x}^2 = \overline{x}_2$, $\overline{y}^2 = \overline{y}_2$), STOP.

objective function as in model (C1) plus constraints (11b).

We will show that the optimal solution of submodel (D1) determines an optimal solution (x, y) for the original model (C1).

$$\min \sum_{m\in M}\sum_{i\in I}\sum_{n\in N}\sum_{j\in J} d_{ij} w_{mn} v_{minj}$$

s. to

$$26x_{1i} + 22x_{2i} + 108x_{3i} + 78x_{4i} \leq 129 \quad \forall i \in \{1, 2\} \tag{D1a}$$

$$x_{m1} + x_{m2} = 1 \quad \forall m \in \{1, \cdots, 4\} \tag{D1b}$$

$$48v_{mi1j} + 79v_{mi2j} + 26v_{mi3j} + 81v_{mi4j} \leq 129x_{mi} \quad \forall m \in \{1, \cdots, 4\},\ i, j \in \{1, 2\} \tag{D1c}$$

$$v_{min1} + v_{min2} = x_{mi} \quad \forall m, n \in \{1, \cdots, 4\},\ i \in \{1, 2\} \tag{D1d}$$

$$\sum_{i\in I} v_{minj} = \sum_{i\in I} v_{m'inj} \quad \forall m, m' \in M : m \neq m', n \in N, j \in J \tag{D1e}$$

$$x_{mi} \in \{0, 1\} \quad \forall m \in \{1, \cdots, 4\},\ i \in \{1, 2\} \tag{D1f}$$

$$v_{minj} \in \{0, 1\} \quad \forall m, n \in \{1, \cdots, 4\},\ i, j \in \{1, 2\}. \tag{D1g}$$

Notice that, from constraints (D1b), the x-variables satisfy (A1). Moreover, from each of constraints (D1a) (and then, $\forall i \in I$) there are two subsets of indices $M'(i)_1, M'(i)_2 \subseteq M$, which variables $x_{m'i(m')}$ can simultaneously take the value 1 $\forall m' \in M'$. In this case, and as the knpasack constraint is the same for $i = 1, 2$, then these subsets also coincide $M'_1 = \{1, 2, 4\}$, $M'_2 = \{3\}$. Thus, taking into account (A1), the set of feasible values for variable x_{mi} $\forall i \in \{1, 2\}$ is given by

$$x_{11} = x_{21} = x_{41} = 1 \quad x_{31} = 0 \tag{D2}$$
$$x_{12} = x_{22} = x_{42} = 0 \quad x_{32} = 1$$

and

$$x_{11} = x_{21} = x_{41} = 0 \quad x_{31} = 1 \tag{D3}$$
$$x_{12} = x_{22} = x_{42} = 1 \quad x_{32} = 0.$$

On the other hand, let us consider the pair $(m, i(m))$ for each $m \in M$, such that $x_{mi(m)} = 1$. In this case, from the constraints (D1d) it follows similar implications as presented in (A3).

Additionally, each pair (m, i) with $i \neq i(m)$, such that $x_{mi} = 0$, implies that $v_{minj} = 0 \ \forall n \in \{1, \cdots, 4\}, j \in \{1, 2\}$. In short, the x, v-variables satisfy the implications given in (A4).

Moreover, from the constraints (D1c) and for each $j \in J$ there are four subsets of indices $N'(j)_1, \cdots N'(j)_4 \subseteq N$, which the variables $v_{mi(m)n'j(n')}$ can simultaneously take the value 1. In this case, as the knapsack constraints have the same R-rhs for $j = 1, 2$, then these subsets coincide, $N'_1 = \{1, 2\}$, $N'_2 = \{3, 4\}$, $N'_3 = \{1, 4\}$, $N'_4 = \{2, 3\}$. And, taking into account (A3), the set of feasible values for the v-variables has to satisfy any of the solution options shown in Tables D1 (options N'_1 and N'_2) and Table D2 (options N'_3 and N'_4).

Table D1. Option N'_1 on the left and Option N'_2 on the right

$v_{mi(m)11} = v_{mi(m)21} = 1$	$v_{mi(m)31} = v_{mi(m)41} = 0$	$v_{mi(m)11} = v_{mi(m)21} = 0$	$v_{mi(m)31} = v_{mi(m)41} = 1$
$v_{mi(m)12} = v_{mi(m)22} = 0$	$v_{mi(m)32} = v_{mi(m)42} = 1$	$v_{mi(m)12} = v_{mi(m)22} = 1$	$v_{mi(m)32} = v_{mi(m)42} = 0$

Table D2. Option N'_3 on the left and Option N'_4 on the right

$v_{mi(m)11} = v_{mi(m)41} = 1$	$v_{mi(m)21} = v_{mi(m)31} = 0,$	$v_{mi(m)11} = v_{mi(m)41} = 0$	$v_{mi(m)21} = v_{mi(m)31} = 1$
$v_{mi(m)12} = v_{mi(m)42} = 0$	$v_{mi(m)22} = v_{mi(m)32} = 1$	$v_{mi(m)12} = v_{mi(m)42} = 1$	$v_{mi(m)22} = v_{mi(m)32} = 0$

From (D2), (D3), (A4) and taking into account the four options for the feasible values of the $v-$variable as shown in Tables D1 and D2, we could present the eight possible combinations, all of them satisfying constraints (D1e). For illustrating the results, we have chosen and shown in Table D3 just one of these combinations, say the $x-$solution expressed as (D2) and the $v-$solution in Option N'_4 as shown in Table D2.

Notice that each column of Table D3 corresponds to one of the possible pairs (n, j), and two terms in the sum of constraint (D1e) appear for each m and each pair of rows ($i \in \{1, 2\}$), Then, it can be checked that the value of $\sum_{i \in I} v_{minj}$ is the same for all m, independently of column (n, j). For instance, if we start with the first column (n, j=(1,1)), it can be seen that

$$v_{1111} + v_{1211} = v_{2111} + v_{2211} = v_{3111} + v_{3211} = v_{4111} + v_{4211} = 0.$$

Table D3. Feasible solutions v for submodel (9) satisfying constraints (D1e)

	$n:1\ j=1$	$j=2$	$n:2\ j=1$	$j=2$	$n:3\ j=1$	$j=2$	$n:4\ j=1$	$j=2$
$m:1\ i=1$	$v_{1111}=0$	$v_{1112}=1$	$v_{1121}=1$	$v_{1122}=0$	$v_{1131}=1$	$v_{1132}=0$	$v_{1141}=0$	$v_{1142}=1$
$i=2$	$v_{1211}=0$	$v_{1212}=0$	$v_{1221}=0$	$v_{1222}=0$	$v_{1231}=0$	$v_{1232}=0$	$v_{1241}=0$	$v_{1242}=0$
$m:2\ i=1$	$v_{2111}=0$	$v_{2112}=1$	$v_{2121}=1$	$v_{2122}=0$	$v_{2131}=1$	$v_{2132}=0$	$v_{2141}=0$	$v_{2142}=1$
$i=2$	$v_{2211}=0$	$v_{2212}=0$	$v_{2221}=0$	$v_{2222}=0$	$v_{2231}=0$	$v_{2232}=0$	$v_{2241}=0$	$v_{2242}=0$
$m:3\ i=1$	$v_{3111}=0$	$v_{3112}=0$	$v_{3121}=0$	$v_{3122}=0$	$v_{3131}=0$	$v_{3132}=0$	$v_{3141}=0$	$v_{3142}=0$
$i=2$	$v_{3211}=0$	$v_{3212}=1$	$v_{3221}=1$	$v_{3222}=0$	$v_{3231}=1$	$v_{3232}=0$	$v_{3241}=0$	$v_{3242}=1$
$m:4\ i=1$	$v_{4111}=0$	$v_{4112}=1$	$v_{4121}=1$	$v_{4122}=0$	$v_{4131}=1$	$v_{4132}=0$	$v_{4141}=0$	$v_{4142}=1$
$i=2$	$v_{4211}=0$	$v_{4212}=0$	$v_{4221}=0$	$v_{4222}=0$	$v_{4231}=0$	$v_{4232}=0$	$v_{4241}=0$	$v_{4242}=0$

The same happens, in the other columns, e.g., the second one (n, j=(1,2)),

$v_{1112} + v_{1212} = v_{2112} + v_{2212} = v_{3112} + v_{3212} = v_{4112} + v_{4212} = 1,$

and, so on. Then, for all $m \in \{1, \cdots, 4\}$, $\sum_{i\in I}, v_{minj}$ is feasible to the knapsack constraint

$$48\sum_{i\in I} v_{mi1j} + 79\sum_{i\in I} v_{mi2j} + 26\sum_{i\in I} v_{mi3j} + 81\sum_{i\in I} v_{mi4j} \leq 129\sum_{i\in I} x_{mi} = 129 \qquad \forall j \in \{1, 2\}$$

Thus, we can write, $\sum_{i\in I} v_{minj} = v_{\bullet\bullet nj} = y_{nj}\ \forall n, j$, where y_{nj} is feasible for the set of constraints of the original model CDAP (C1) and it is the optimal solution when the corresponding v does it. Notice that one of the optimal solutions, say, (x, y) of that model is given by the $x-$vector expressed as (D2) and the $y-$vector expressed as

$$\begin{aligned} y_{11} = y_{41} = 0 \quad y_{21} = y_{31} = 1 \\ y_{12} = y_{42} = 1 \quad y_{22} = y_{32} = 0 \end{aligned} \tag{D4}$$

in the solution chosen in Table D3. (The other optimal solution is given by the $x-$vector expressed as (D3) and the y-vector as implied by the v-solution vector in Option N_2' as shown in Table D1).

Finally, notice that when constraint (D1e) is not satisfied, then the solution under consideration (x, v) provides just a lower bound for the optimal solution value of model (C1).