Algorithms for the Cross-dock Door Assignment Problem

Monique Guignard Peter M. Hahn Artur A. Pessoa
Daniel Cardoso da Silva

Abstract

In a cross-dock facility, goods are moved by forklift from incoming truck platforms (strip doors) to temporary holding areas and then to outgoing truck platforms (stack doors) or directly from strip doors to stack doors. Costs within the cross-dock may be minimized by appropriate assignment of strip doors to incoming trucks and stack doors to outgoing trucks, thus minimizing the distances that forklifts must travel. Optimizing strip and stack door assignments given the shape of the cross-dock and the origin-destination volumes of goods is known as the Cross-dock Door Assignment Problem (CDAP). The CDAP is very difficult for exact methods because of its quadratic objective function. We propose two heuristics for this problem, one ad-hoc using local search, and the other a metaheuristic called CH (for Convex Hull), designed explicitly for quadratic 0-1 problems with linear constraints. We then compare the proposed heuristics with one exact solution method for optimizing door assignments at typical cross-dock layouts having as many as 20 doors, 20 origins and 20 destinations and randomly generated origin-destination volumes of trucked goods.

1 Introduction

1.1 Background and Literature Review

Cross-docking is a logistics technique used in the retail and trucking industries to rapidly consolidate shipments from disparate sources and realize economies of scale in outbound transportation. Cross-docking essentially eliminates the costly inventory-holding functions of a warehouse, while still allowing it to serve its consolidation and shipping functions. The idea is to transfer shipments directly from incoming to outgoing truck trailers, without storage in between. Shipments typically spend less than 24 hours in a cross-dock, sometimes less than an hour. With the process of moving shipments from the receiving dock (strip door) to the shipping dock (stack door), bypassing storage, cross-docking reduces inventory carrying cost, transportation cost, and other costs associated with material handling. Research topics on the design and operational problems in cross-docking include the size of the cross-dock, the number of strip and stack doors, the width of the cross-dock, shapes for different cross-dock sizes, and assigning strip and stack doors to incoming and outgoing truck trailers, which is addressed here.

In a cross-dock facility, goods are moved by forklift from the incoming strip door to the appropriate outgoing stack door, for a specific destination. The dynamic nature of the flow patterns between origin-destination combinations makes the assignment of origins to strip doors and assignment of destinations to stack doors a difficult combinatorial optimization problem.

The cross-docking problem is a type of assignment problem for which many studies have been reported in the literature. Peck [15] developed a simulation for the integer programming model of the trailer-to-door assignments that minimizes the total transfer time. Tsui and Chang [18, 19] presented a general model of the dock door assignments and then developed a solution based on branch and bound. Kinnear [12] explained the advantages afforded by cross-docking. Gue [7] addressed the cross-dock facility layout as the arrangement of strip/stack doors and

the assignment of destination trucks to stack doors. Gue proposed a look-ahead scheduling algorithm to reduce more labor cost compared to first-come first-served policy. Bartholdi and Gue [3] described models that guide a local search routine in cross-dock door assignment so as to minimize the total labor cost. Their layout models balanced the cost of moving freight from incoming trailer to outgoing trailers with the cost of delays due to different types of congestion. Bermudez [5] developed a genetic algorithm for assigning doors in order to minimize the total weighted travel distance. Sung and Song [17] designed an integrated service network for a cross-docking supply chain network. Lim et al. [13] minimized the total shipping distance of freight inside a cross-dock facility. Miao et al. [14] aimed to find an optimal assignment of doors and scheduling of trucks that minimizes the operational cost of the cargo shipments and the total number of unfulfilled shipments at the same time. Bozer and Carlo [6] proposed a simulated annealing heuristic to determine the door assignments in cross-docks, which is formulated as a Quadratic Assignment Problem (QAP) with rectilinear distances.

All previous works assumed that trucks are aggregated so that each group of incoming (outgoing) trucks exactly fits into one strip (stack) door. Zhu et al. [21] proposed a new model for the CDAP the generalizes the previous one allowing trucks to be aggregated in a more natural way, e.g. by origins (destinations) in the case of incoming (outgoing) trucks. Their model handles the situation where the strip door serves more than one origin and the stack door serves more than one destination. From now on, let us refer to this generalized problem simply as CDAP. In [21], Zhu et al. also apply the formulation of the Generalized Quadratic 3-dimensional Assignment Problem (GQ3AP) to devise an exact branch-and-bound algorithm for CDAP. Their method was only tested with one instance having 8 origins, 8 destinations, 4 strip doors, and 4 stack doors, varying the door capacities. Based on its mathematical formulation, the CDAP could be solved as a GQAP [10].

1.2 Zhu's Cross-dock Door Assignment Problem

Here is Zhu's mathematical model of the Cross-dock Door Assignment Problem (CDAP).
Parameters:

M number of origins,

N number of destinations,

I number of strip doors,

J number of stack doors,

w_{mn} number of trips required by the material handling equipment to move items originating from m to the cross-dock door where freight destined for n is being consolidated,

d_{ij} distance between strip door i and stack door j,

s_m volume of goods from origin m,

S_i capacity of strip door i,

r_n demand from destination n,

R_j capacity of stack door j.

Decision Variables:

$x_{mi} = 1$ if origin m is assigned to strip door i, $x_{mi} = 0$ otherwise,

$y_{nj} = 1$ if destination n is assigned to stack door j, $y_{nj} = 0$ otherwise.

Formulation:

$$\text{Minimize:} \quad \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{m=1}^{M}\sum_{n=1}^{N} d_{ij} w_{mn} x_{mi} y_{nj} \tag{1}$$

$$\text{Subject to:} \quad \sum_{m=1}^{M} s_m x_{mi} \leq S_i, \qquad i = 1, 2, ..., I \tag{2}$$

$$\sum_{i=1}^{I} x_{mi} = 1, \qquad m = 1, 2, ..., M \tag{3}$$

$$\sum_{n=1}^{N} r_n y_{nj} \leq R_j, \qquad j = 1, 2, ..., J \tag{4}$$

$$\sum_{j=1}^{J} y_{nj} = 1, \qquad n = 1, 2, ..., N \tag{5}$$

$$x_{mi} = 0 \text{ or } 1 \qquad m = 1, 2, ..., M,\ i = 1, 2, ..., I$$

$$y_{nj} = 0 \text{ or } 1 \qquad n = 1, 2, ..., N,\ j = 1, 2, ..., J$$

Here, (2) makes sure that the capacity of each strip door S_i is not exceeded, (3) makes sure that each origin gets assigned only one receiving (strip) door, (4) makes sure that the capacity of each outbound (stack) door R_j is not exceeded, and (5) makes sure that each destination is assigned only one stack door.

1.3 Contribution and Paper Organization

The CDAP (even considering the previous model of Tsui and Chang) is very difficult for exact methods because of its quadratic objective function. Besides, if the cost function is replaced by a linear one, the resulting problem, although still NP-hard, can be solved much more efficiently using commercial Mixed Integer Programming (MIP) solvers. Motivated by this observation, we propose two heuristics for the CDAP, both using the exact methods for similar subproblems with linear costs instead of quadratic. The first method is a multi-start local search that alternates between optimally reassigning only incoming trucks and only outgoing trucks. The second one uses a more sophisticated technique called Convex Hull Relaxation (CHR) to linearize the quadratic CDAP function. It alternates between using a linear approximation of the cost function at the current solution and minimizing the quadratic cost function over an expanding inner approximation of the integer feasible region. In order to test the proposed heuristics, we generate a more complete set of benchmark instances varying not only the door capacities but also the number of origins/destinations and doors. The new set of instances use typical cross-dock layouts with up to 20 doors, and randomly generated origin-destination volumes of trucked goods. We then present experimental results comparing the two heuristics and the algorithm of Zhu et al.

In Section 2, we describe our multi-start local search heuristic. In Section 3, we show how the CDAP can be solved approximately, using CHR. In Section 4, we explain how the set of bechmark instances has been generated and present our experimental results. Finally, our conclusions are presented in Section 5.

2 First Heuristic method: Local Search by MIP

2.1 Reducing a part of the CDAP as a GAP

Ross and Soland [16] defined the Generalized Assignment Problem (GAP) as the most basic version of the assignment problems that allow an agent to be assigned to multiple tasks. Its importance derives not only from its direct application, but also from the fact that it appears as

a subproblem in many other practical and more complex problems on the literature. The GAP can be formulated as follows:

$$\text{Minimize:} \quad \sum_{m=1}^{M} \sum_{i=1}^{I} C_{im} z_{im} \tag{6}$$

$$\text{Subject to:} \quad \sum_{m=1}^{M} a_{im} z_{im} \leq b_i, \quad i = 1, 2, ..., I \tag{7}$$

$$\sum_{i=1}^{I} u_{im} z_{im} = 1, \quad m = 1, 2, ..., M \tag{8}$$

$$z_{im} = 0 \text{ or } 1 \qquad i = 1, 2, ..., I, \; m = 1, 2, ..., M \tag{9}$$

where $z_{im} = 1$ if agent i is assigned to task m, 0 otherwise, c_{im} is the cost of assigning agent i to task m, a_{im} is the amount of capacity employed by agent i to execute task j, and b_i is the total capacity available for agent i. The first set of constraints ensures that the set of tasks assigned to an agent does not exceed its capacity and the second set of constraints ensures that every task is assigned to only one agent.

In the CDAP formulation in Section 2, the decision variables and the constraints are separately defined for each side of the cross-dock. While x_{mi} variables define the assignment of each origin to a strip door and constraints (2) and (3) makes sure that the capacities of strip doors were not exceeded and origins get assigned only once, respectively, on the inbound side of the cross-dock, the variables and constraints (4) and (5) have the same purpose to the outbound side.

Assuming a fixed assignment of all destinations to stack doors, i.e. fixed values for the variables y_{nj}, the restricted CDAP can solved as a GAP as follows:

(a) each agent corresponds to a strip door i, for $i = 1, 2, \ldots, I$;

(b) each task corresponds to an origin m, for $m = 1, 2, \ldots, M$;

(c) set $a_{im} = s_m$ and $b_i = S_i$, for $i = 1, 2, \ldots, I$, and $m = 1, 2, \ldots, M$;

(d) set $c_{im} = \sum_{n=1}^{N} \sum_{j=1}^{J} d_{ij} w_{mn} y_{nj}$, for $i = 1, 2, \ldots, I$, and $m = 1, 2, \ldots, M$;

(e) solve the resulting GAP instance defined by (6)-(9);

(e) set $x_{mi} = z_{im}$, for $i = 1, 2, \ldots, I$, and $m = 1, 2, \ldots, M$.

An analogous reduction can be done if the value of all x_{mi} variables are fixed and one wants to optimize the y_{nj} variables.

2.2 Heuristic

In this section, we describe how the GAP reduction is used in a local search method, in order to find good solutions for the CDAP. The heuristic has the following steps:

Step 1: Generate a random solution for the inbound side of the CDAP (x_{mi})

- For each origin m, randomly select a door i from the doors that are capable to assign origin m.
- Assign origin m to the selected door i and decrease the available capacity of the door.
- If there is no door capable to attend an origin m, then Step 1 is restarted.

Step 2: With the value of x_{mi} variables found on Step 1, generate a GAP for the outbound side of the CDAP and solve it. Keep the objective function value and the y_{nj} variable values.

Step 3: With the value of y_{nj} variables found on Step 2, generate a GAP for the inbound side of the CDAP and solve it. If the objective function value found is better than the value found on step 2, keep it and the x_{mi} variable values, otherwise stop.

Step 4: Repeat step 3 alternating the side of cross-dock being optimized until the objective function value does not improve. Keep the best objective function value found and the corresponding x_{mi} and y_{nj} values.

In order to avoid getting stuck on a local optimum, steps 1 to 4 are repeated one hundred times and the best solution is selected.

The previous version of the heuristic is referred to as LS1. Another variant of this method, denoted by LS2, is also tested. It differs from LS1 only at Step 1. Instead of generating a random solution that necessarily respects the door capacities, it randomly assigns each origin m any strip door i. This variant is designed to save processing time in the generation of initial solutions, specially for instances where the door capacities are tight. The modified Step 1 does not affect feasibility of the final solution since all origins are reassigned at Step 3, this time necessarily respecting the capacity restrictions.

3 Second Heuristic method: The CH heuristic

The convex hull (CH) heuristic is a multi-start metaheuristic, designed for pure integer nonlinear optimization problems with linear constraints. It is generic, in the sense that it can be applied to any such nonlinear problem as long as linear problems subject to the same constraints are by comparison, much easier to solve, at least approximately. A single start of the algorithm, loosely based on simplicial decomposition [20], alternates between a continuous nonlinear problem with one linear constraint, whose number of variables increases by one at each iteration, and one linear MIP problem subject to all original constraints. It can make use of software features such as CPLEX's solution pool to enlarge – and enhance the quality of – the sample of integer feasible solutions found.

[2] and [1] independently introduced a relaxation method called convex hull relaxation (CHR), an extension of the primal relaxation of [8], for computing both a lower bound on the optimal value of a nonlinear integer minimization program, and good integer feasible solutions. This can be defined for both convex and nonconvex problems, however it is only computationally meaningful for convex problems. Indeed, in the nonconvex case, the algorithm produces a value that is in general not a valid bound, yet it still generates, as a by-product, what are almost always high quality integer feasible solutions. The main difference computationally with the convex case is that the convergence criterion (adapted from simplicial decomposition) is usually satisfied very quickly, necessitating a restart from a different initial step if one wants to generate a solution sample of reasonably large size. We will refer to this approach as the convex hull heuristic, CH. Notice that in the convex case, the CH heuristic is by default single-start, since the algorithm always converges to the same optimal value. Yet it might be possible to make it multi-start as the sequence of extreme points generated may depend on the starting point. This has no bearing on our study, however, as the cost matrix is in general not positive semidefinite. In what follows, we will then apply the multi-start heuristic version of the algorithm to the CDAP model described in subsection 1.2. A comparison of runtimes and solution quality using exact solvers, the ad-hoc local search heuristic and the convex hull heuristic is given in Section 7.

We now describe the CH heuristic in greater detail.

Consider the nonlinear integer programming program (NLIP)

(NLIP)

$$\begin{gathered}\min_{\mathrm{y}} f(y) \\ s.t.\ Ay \leq b \\ y \epsilon Y\end{gathered}$$

where *f(y)* is a nonlinear differentiable function of y,
A is an $m \times n$ constraint matrix,
b is a resource vector in R^m,
Y is a subset of R^n specifying integrality restrictions on y.

The convex hull relaxation (CHR) of NLIP is:
(CHR)

$$\begin{gathered}\min_{\mathrm{y}} f(y) \\ s.t.\ y \epsilon Conv\left\{x | Ax \leq b, x \epsilon Y\right\}\end{gathered}$$

It is easy to see that (CHR) is indeed a relaxation of (NLIP). We will now describe the k^{th} iteration of the CH heuristic for a given start. Let us define the Convex Hull Subproblem (CHS) at the k^{th} iteration as
(CHS)

$$\begin{gathered}\min_{\mathrm{y}} \bigtriangledown f(x^{(k)})^T (y - x^{(k)}) \\ s.t.\ y \epsilon Conv\left\{x | Ax \leq b, x \epsilon Y\right\}\end{gathered}$$

where $x^{(k)}$ is a feasible point for CHR, i.e., in Conv $\{x|Ax \leq b, x \epsilon Y\}$.

$x^{(k)}$ is the current linearizing point, i.e., at $x^{(k)}$ one approximates the original quadratic function $f(y)$ by its first order approximation $H(y)$:

$$H(y) = f(x^{(k)}) + \bigtriangledown f(x^{(k)})^T (y - x^{(k)}).$$

Note that CHS is a linear program. As such it has an equivalent integer program, the Integer Programming Subproblem (IPS):
(IPS)

$$\begin{gathered}\min_{\mathrm{y}} \bigtriangledown f(x^{(k)})^T (y - x^{(k)}) \\ s.t.\ Ay \leq b \\ y \epsilon Y.\end{gathered} \tag{10}$$

Let $y^{(k)}$ be the optimal solution of (IPS). $y^{(k)}$ depends on $x^{(k)}$ because the linear function $H(x)$ depends on the linearization point $x^{(k)}$. Because the objective function is linear, $y^{(k)}$ also solves problem (CHS). At each iteration, one solves an LIP with a different linearized function $H(x)$, while the constraint set is always the same.

In the convex case, the solution to (IPS), $y^{(k)}$, is a new extreme point of the convex hull, unless $x^{(k)}$ is optimal for the convex hull relaxation problem (CHR). In that case the algorithm stops, because no new extreme point is generated, which means that the set $Conv\left\{x_0, y^{(1)}, y^{(2)}, ...\right\}$ already contains the global optimum. If the objective function is nonconvex, the algorithm may stop even when $x^{(k)}$ is not the global optimum. In both cases, if $y(k)$ is a new extreme point, it is used to expand the area the program will search in the next iteration, as the set $Conv\left\{x^{(0)}, y^{(1)}, y^{(2)}, ...\right\}$ will expand and give a better approximation of the integer convex hull. Until the algorithm stops, each iteration produces a new vertex $y^{(k)}$ of the integer convex hull and then minimizes the original nonlinear objective function over $Conv\left\{x^{(0)}, y^{(1)}, y^{(2)}, ...\right\}$.

This minimization can be written as a nonlinear continuous problem over a set of weights allocated to each extreme point. The only constraints are that these weights are nonnegative and add up to 1. There is one more weight to consider at each iteration, and the maximum number of weights is the number of iterations plus 1. The solution of the nonlinear continuous problem is the new linearization point, $x^{(k+1)}$.

If $f(y)$ is convex, the algorithm will converge after a finite number p of iterations to a point $x^{(p)}$ in $Conv\{y|Ay \leq b, y\epsilon Y\}$. In the worst case, the algorithm will generate all extreme points of the integer convex hull. Convergence happens when $f(x^{(p-1)}) = f(x^{(p)})$, i.e., if the latest point $x^{(p)}$ has not improved the nonlinear optimal value. In the convex case, $f(x^{(p)})$ is a valid lower bound on the integer optimum, because

$$f(x^{(p)}) = min\{f(y)|y\epsilon Conv\{Ay \leq b, y\epsilon Y\}\}.$$

A by-product of CHR is that it produces a number of feasible integer solutions, $y^{(1)}$, ..., $y^{(p)}$, and we can compute $f(y^{(1)})$, $f(y^{(2)})$, ..., $f(y^{(p)})$ and keep the best point, call it $y*$, as a best feasible value. Thus even in the nonconvex case, CHR works also as a primal heuristic. In the convex case, CHR produces both a valid bound on the optimum, $f(x^{(p)})$, and a best found feasible integer solution $y*$. In the nonconvex case, it only produces a best feasible integer solution.

We allow two changes over a standard implementation of simplicial decomposition. First it is not essential in the nonconvex case to solve every single intermediate problem (IPS) to optimality. It does happen in practice that out of maybe fifty or sixty calls to the linear MIP solver, one or two problems turn out to require substantially more time. To avoid this erratic behavior, we impose an upper bound on the runtime of every (IPS) ten or 20% larger than the average (IPS) runtime (one cannot do this in the convex case as this might affect the overall convergence of the algorithm). It might however affect the final best value, positively or negatively, but keeps the time manageable. The second deviation is based on the solution pool feature of CPLEX. While solving (IPS) using commercial software, the BB code searches for a best feasible value for the $linearized$ objective function. Successive incumbents can also be tested on the fly in terms of another criterion, in our case their value for the $original$ nonlinear function, and the best solution and its value are recorded. In many of our experiments, the best value obtained was actually coming from the solution pool of CPLEX.

The performance of the CHR heuristic for solving CDAP is given below.

4 Experimental results

4.1 Input data

4.1.1 Flow matrix

Since it is very difficult to get complete sets of data from commercial shippers, we had to generate plausible datasets similar to the partial ones we obtained in order to test the algorithms discussed in this paper. We needed to create a flow matrix based on the given demands of each customer. The following process was developed to generate an origin-destination flow matrix F.

We assume that each destination n will receive goods from at least one origin m and each origin m will send goods for at least one destination n. First, for each origin m, one destination n was randomly chosen and the demand from the corresponding origin w_{mn} is given by a random integer between 10 and 50. A similar process was performed for each destination n which has not been drawn in the above process, giving a random integer between 10 and 50 to a randomly assigned origin m. A third step was made by choosing a random origin-destination flow w_{mn} and in case this flow has not been assigned before, give it a random integer between 10 and 50. The third step was repeated until at least 25% of the flow matrix has been filled.

All instances have been generated with $M = N$, following the procedure described above.

4.1.2 Distance matrix

According to Bartholdi and Gue [4], most cross-docks, specially the smaller ones, are long, narrow rectangles. This I-shaped design offers the chance to move freight directly across the dock, reducing handling costs and time of the operation. On most cross-docks the doors are equally spaced and generally with a 12-foot offset, but the width of the cross-dock depends on the estimated need of the operation and in the LTL trucking industry usually are 60-120 feet wide.

Figure 1 shows a simple layout of a cross-dock, which has a long rectangular shape, with strip doors on the left side and stack doors on the right side.

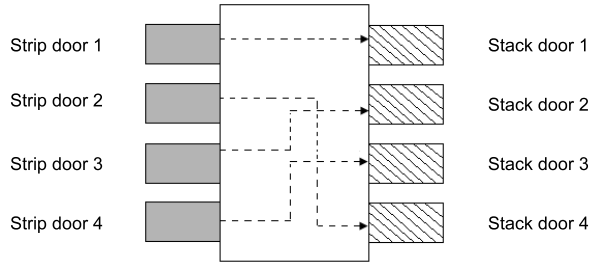

Figure 1: A simple cross-dock layout

Considering an average cross-dock width of 90 feet and doors with 12 feet, we have an approximate proportion of 8 to 1. So, a typical set of distances between the five strip doors and five stack doors are shown in the distance matrix D, where d_{ij} represents the distance from strip door i to stack door j.

$$D = \begin{pmatrix} 8 & 9 & 10 & 11 & 12 \\ 9 & 8 & 9 & 10 & 11 \\ 10 & 9 & 8 & 9 & 10 \\ 11 & 10 & 9 & 8 & 9 \\ 12 & 11 & 10 & 9 & 8 \end{pmatrix}$$

All instances have been generated with $I = J$, with values of d_{ij} always ranging from 8 to $8 + I - 1$ following the same pattern above.

4.1.3 Supplies of the origins and demands of the destinations

Once the origin-destination flow is generated, the volume of goods from each origin m is calculated as $s_m = \sum_{n=1}^{N} w_{mn}$. Similarly, the demand from each destination n is calculated as $r_n = \sum_{m=1}^{M} w_{mn}$. To complete the input data for the CDAP problem, the capacity of each strip door is given an identical value, which is calculated by dividing the total volume of goods from all origins by the total number of strip doors, and then adding a capacity slack to the quotient. We use slacks of 5%, 10%, 15%, 20% and 30% for the strip door capacities. A similar procedure is used to generate (identical) stack door capacities: divide the total demand from all destinations by the total number of stack doors, and then add a capacity slack to the quotient which corresponds to the same percentage used for the strip doors.

4.2 Results

Table 1 shows the results of the three proposed methods for the instances generated, as described in Subsection 4.1.

Table 1: Experimental Results

	LS1 heur.		**LS2 heur.**		**CH heur.**		**B&B**	
Instance	Best soln.	Proc. (sec.)	Best soln.	Time (sec.)	Best soln.	Time (sec.)	Opti-mum	Time to opt.
8x4S30	5063	2	5063	2	5063	17	5063	0
8x4S20	5086	2	5086	3	5086	17	5086	0
8x4S15	5112	2	5112	2	5112	9	5112	0
8x4S10	5169	2	5169	3	5169	14	5169	0
8x4S5	5174	1	5174	2	5174	13	5174	0
9x4S30	5904	2	5904	2	5904	14	5904	0
9x4S20	5937	2	5937	2	5937	19	5937	0
9x4S15	5976	3	5976	3	5976	21	5976	0
9x4S10	6027	2	6027	3	6027	17	6027	0
9x4S5	6047	2	6047	3	6047	15	6047	0
10x4S30	6193	3	6193	3	6193	19	6193	0
10x4S20	6267	3	6267	4	6267	26	6267	0
10x4S15	6296	3	6296	3	6296	28	6296	0
10x4S10	6325	3	6325	3	6325	30	6325	0
10x4S5	6518	5	6518	6	6518	16	6518	2
10x5S30	6324	3	6333	4	6333	18	**6308**	0
10x5S20	6374	3	6363	4	6354	18	**6342**	0
10x5S15	6397	3	6397	4	6397	16	6397	0
10x5S10	6476	4	6476	5	6476	23	6476	0
10x5S5	6616	6	6616	6	6616	13	6616	7
11x5S30	7428	4	7424	5	7420	18	7420	15
11x5S20	7465	5	7465	5	7478	17	**7439**	14
11x5S15	7542	5	7542	5	7542	20	**7535**	126
11x5S10	7572	5	7572	6	7572	18	7572	33
11x5S5	7812	12	7812	12	7812	16	7812	731
12x5S30	7942	6	7925	6	7925	21	**7923**	21
12x5S20	7939	5	7939	6	7939	18	7939	32
12x5S15	7939	4	7939	6	7939	18	7939	9
12x5S10	7978	4	7988	5	7991	15	7978	16
12x5S5	8072	5	8072	7	8072	13	8072	68

Experimental Results (cont.)

	Heur. 1		Heur.2		CHR heur.		B&B	
Instance	Best soln.	Proc. (sec.)	Best soln.	Proc. (sec.)	Best soln.	Proc. (sec.)	Opti-mum	Time (sec.) to opt.
12x6S30	10228	6	10228	7	10277	20	10228	97
12x6S20	10331	7	10339	6	10323	16	**10312**	201
12x6S15	10395	5	10420	6	10374	17	**10362**	198
12x6S10	10475	6	10496	7	10480	14	**10456**	98
12x6S5	10891	10	10894	14	10894	14	10891	7,921
15x6S30	13567	10	13567	10	13626	16	13567	1,214
15x6S20	13769	10	13754	11	13720	17	13720	3,422
15x6S15	13769	9	13799	12	13765	18	13765	225
15x6S10	13856	11	13852	11	13872	15	**13803**	1,197
15x6S5	13960	13	13927	16	13983	18	13927	31,091
15x7S30	14415	13	14423	13	14414	19		
15x7S20	14596	14	14533	16	14533	19	**14514**	unproven
15x7S15	14678	13	14680	14	14678	22		
15x7S10	14841	15	14818	17	14810	22		
15x7S5	15054	25	15054	25	15096	26		
20x10S30	28800	39	28541	41	28571	62		
20x10S20	29130	38	28992	40	29089	63		
20x10S15	29278	35	29184	37	29529	65		
20x10S10	29286	36	29286	36	29498	52		
20x10S5	29945	254	30033	45	29933	87		

For the CDAP heuristics described in this paper, Table 1 presents the best solution found by each instance and the corresponding processing time in seconds needed to find that solution. In order to demonstrate the worth of the three heuristics, two columns were added, continuing respectively, the optimum for each problem (if it could be found by a branch and bound search) and the time it took for the branch and bound search to find the first optimal solution (there are sometimes several). We chose our Generalized Quadratic 3-dimensional Assignment Code solver [21] to find optimal solutions for the CDAP test instances presented in this paper.

The runtime results presented in Table 1 for Heuristics LS1 and LS2 were recorded on a single cpu of an AMD Phenom 9600 2.31GHz Quad-Core Processor. The operating system for those heuristics was Windows XP Professional Version 2002, Service Pack 2 and the version of CPLEX was 12.1. The runtime results presented for the CH heuristic were recorded on a single cpu of a 3.00GHz Quad-Core Intel Xeon CPU E5450 Processor on a Dell PowerEdge 2950 Linux server. The runs were made using GAMS version 28.2 and times include GAMS overhead.

For the forty instances whose optimum solutions were known, in only nine cases was the optimum solution not found. In those nine cases, the worst gap between solution found and optimum was only 1/4 of one percent, a negligible amount. In addition, there are negligible differences between the three heuristic implementations.

For problem instances smaller than 11x5 (11 origins, 11 destinations, 5 strip doors and 5 stack doors), the branch and bound search is as efficient or possibly more efficient than any of the heuristics. But, this is quickly overshadowed by what happens for larger instances. In fact, for instances larger than 12x6 (with the exception of the 15x7S20 instance), the GQ3AP solver gave up searching for optimum solutions after running168 hours (one week) on a single Opteron 2.3 Ghz cpu of a Dell PowerEdge 1950/SunFire X2200 Cluster at Clemson University. All of the branch-and-bound runtimes reported in Table 1 were measured on that computing cluster. Regarding the single instance where a solution was found in the process of attempting to solve the 15x7S20 instance, that solution is indeed better than any of the solutions found

by the heuristics. However, since the branch-and-bound search was not completed, it could not be verified as optimal. Clearly, there is need for additional development for finding exact solutions for the unsolved instances in the table. This work would, however, require much more time than was available for the writing of this paper.

5 Conclusions

The CDAP is very difficult for exact methods because of its quadratic objective function. We propose two heuristics for this problem, one ad-hoc using local search, and the other a meta-heuristic called CH (for Convex Hull), designed explicitly for quadratic 0-1 problems with linear constraints. We then compare the proposed heuristics with one exact solution method for optimizing door assignments at typical cross-dock layouts having as many as 20 doors, 20 origins and 20 destinations and randomly generated origin-destination volumes of trucked goods.

Data sets of CDAP for different solvers were generated and instances of different problem sizes were tested with the above-mentioned algorithms. It was found that the input data of CDAP should be modified to adapt the GQ3AP model, which was initially reported by Guignard et al. [9].

Computational experiments were conducted by implementing the existing algorithmic tools. Existing codes were used to solve CDAP test instances exactly using a GQ3AP algorithm. Also, CDAP input data for the CHR heuristic was generated for testing the CHR heuristic algorithm by others. As expected, the runtimes of the exact algorithm grow exponentially when problem size grows. Recommendations regarding the use of the tested algorithms were made, based on the results.

Acknowledgments

The material is partially based upon work supported by the U.S. National Science Foundation, under Grant No. DMI-0400155.

The branch-and-bound results reported herein were made possible through the advanced computing resources deployed and maintained by Clemson Computing and Information Technology and the Clemson University International Center for Automotive Research Center for Computational Mobility Systems.

References

[1] A. Ahlatioglu and M. Guignard. The convex hull relaxation for nonlinear integer programs with linear constraints. Technical report, University of Pennsylvania, The Wharton School, OPIM Department, 2007. latest revision January 2009.

[2] V. Albornoz. *Diseno de Modelos y Algoritmos de Optimizacion Robusta y su Aplicacion a la Planificacion Agregada de la Produccion*. PhD thesis, Universidad Catolica de Chile, Santiago, Chile, 1998.

[3] J. J. Bartholdi and K. R. Gue. Reducing labor costs in an LTL cross-docking terminal. *Operations Research*, 48(6):823–832, 2000.

[4] J. J. Bartholdi and K. R. Gue. The best shape for a crossdock. *Transportation Science*, 38(2):235–244, 2004.

[5] R. A. Bermudez. A genetic algorithm approach to LTL break-bulk terminal door assignment. Master's thesis, University of Arkansas, Fayetteville, AR 72701, 2002.

[6] Y. A. Bozer and H. J. Carlo. Optimizing inbound and outbound door assignments in less-than-truckload crossdocks. *IIE Transactions*, 40(11):1007–1018, 2008.

[7] K. R. Gue. The effects of trailer scheduling on the layout of freight terminals. *Transportation Science*, 33(4):419–428, 1999.

[8] M. Guignard. Primal relaxations for integer programming. In *Proceedings of the VII CLAIO*, Santiago, Chile, 1994.

[9] M. Guignard, P. M. Hahn, Z. Ding, B.-J. Kim, , and Y.-R. Zhu. Extensions and variations on quadratic assignment problems including the quadratic 3-dimensional assignment problem (q3ap). In *Proceedings of 2006 NSF Design, Service, and Manufacturing Grantees and Research Conference*, St. Louis, MO, 2006.

[10] P. M. Hahn. A discourse on the relationship between the cdap and the gqap. Private Communication, University of Pennsylvania, ESE Department, 2006.

[11] P. M. Hahn, B. J. Kim, M. Guignard, J. M. Smith, and Y.-R. Zhu. An algorithm for the generalized quadratic assignment problem. *Computational Optimization and Application*, 40(3):351–372, 2008.

[12] E. Kinnear. Is there any magic in cross-docking? *Supply Chain Management: An International Journal*, 2(2):49–52, 1997.

[13] A. Lim, H. Ma, and Z. Miao. Truck dock assignment problem with time windows and capacity constraint in transshipment network through crossdocks. In *Computational Science and Its Applications: ICCSA 2006*, pages 688–697, Glasgow, United Kingdom, 2006.

[14] Z. Miao, A. Lim, and H. Ma. Truck dock assignment problem with operational time constraint within crossdocks. *European Journal of Operational Research*, 4031:262–271, 2006.

[15] K. E. Peck. *Operational analysis of freight terminals handling less than container load shipments*. PhD thesis, University of Illinois at Urbana-Champaign, Urbana, IL 61801, 1983.

[16] G. T. Ross and R. M. Soland. A branch and bound algorithm for the generalized assignment problem. *Mathematical Programming*, 8:91–103, 1975.

[17] C. S. Sung and S. H. Song. Integrated service network design for a cross-docking supply chain network. *The Journal of the Operational Research Society*, 54(12):1283–1295, 2003.

[18] L. Y. Tsui and C.-H. Chang. A microcomputer based decision support tool for assigning dock doors in freight yards. *Computers and Industrial Engineering*, 19(1-4):309–312, 1990.

[19] L. Y. Tsui and C.-H Chang. An optimal solution to a dock door assignment problem. *Computers and Industrial Engineering*, 23(1-4):283–286, 1992.

[20] Balder Von Hohenbalken. Simplicial decomposition in nonlinear programming algorithms. *Mathematical Programming*, 13:49–68, 1977.

[21] Y.-R. Zhu, P. M. Hahn, Y. Liu, and M. Guignard. New approach for the cross-dock door assignment problem. In *Anais do XLI Simpósio Brasileiro de Pesquisa Operacional*, Porto Seguro, Bahia, Brazil, 2009.