Finding Alternative Solutions of Interest for the Cross-Dock Door Assignment (CDAP) Problem

Monique Guignard and Steven O. Kimbrough

University of Pennsylvania

Abstract. We explore the problem of discovering interesting solutions ("solutions of interest" or SoIs) in the context of the cross-dock door assignment problem (CDAP), a challenging integer programming problem that is quadratic in the objective. These solutions of interest may be optimal or not, but potentially valuable for sensitivity analysis and other post-solution tasks. We compare the CHH (convex hull heuristic), a mathematical programming metaheuristic, with the FI-2Pop GA+SOI, an evolutionary computation metaheuristic.

Keywords: cross-dock door assignment problem, CDAP, convex hull heuristic, CHH, feasible-infeasible two-population genetic algorithm, solutions of interest, multiple solutions

1 Introduction

Research in conventional optimization is largely focused on the problem of finding better solution methods, whether exact or heuristic, for optimization models, and on assessing and comparing the performance of the solution methods that are available. We explore a different problem in this paper. We informally define a *solution of interest* (SoI) as a possibly non-optimal and possibly infeasible solution that is nevertheless likely to be relevant to decision making informed by the model. Concretely, we identify SoIs as either feasible with comparatively good objective values (comparing to the optimum or best available) and comparatively high levels of slack in the constraints, or as infeasible with superior objective values and modest levels of constraint violation. (See [9] for a more extended discussion of SoIs.) We explore the cross-dock door assignment problem (CDAP) with the aim of finding SoIs. In doing so, we contrast and compare two metaheuristic approaches, one based on a variety of evolutionary computation, the other based on mathematical programming. We begin, in the next section, by describing the CDAP.

2 The Cross-Dock Door Assignment Problem (CDAP)

In the CDAP we assume a cross-docking facility with I incoming (strip) doors and J outgoing (stack) doors, with M origins (sources) and N destinations (sinks). Trucks with goods from various sources and for various destinations arrive at the cross-dock and are assigned a strip door, at which the trucks are unloaded (stripped) and their goods sorted and moved to trucks docked at the stack doors. The problem is to position the incoming trucks and the outgoing trucks so that the total distance the goods (and the people who push the loaded carts) have to travel across the cross-dock is minimized. Each box has a specified destination and must be loaded in an outbound truck going to that destination.

We illustrate with an example, the SetA_8x4S30 benchmark CDAP from [5]. To begin there is the matrix/array W:

$$W = \begin{matrix} 0 & 0 & 26 & 0 & 0 & 0 & 0 & 0 \\ 22 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 41 & 32 & 0 & 50 & 30 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 & 31 & 0 & 0 \\ 40 & 0 & 0 & 44 & 0 & 0 & 50 & 0 \\ 0 & 47 & 0 & 31 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 43 & 0 & 0 & 0 \\ 0 & 44 & 31 & 0 & 0 & 0 & 0 & 31 \end{matrix} \tag{1}$$

The interpretation is that w_{mn} is the amount of goods (think: cubic feet) in the incoming truck that must be transported from source m to destination n. Thus, in this example, incoming truck #6 is loaded with 78 units of goods from source 6, of which 47 units need to be shipped (loaded on a truck for) destination 2 and 31 need to be shipped to destination 4. W is 8×8, so there are 8 distinct sources and 8 distinct destinations.

We also have the matrix/array D:

$$D = \begin{matrix} 8 & 9 & 10 & 11 \\ 9 & 8 & 9 & 10 \\ 10 & 9 & 8 & 9 \\ 11 & 10 & 9 & 8 \end{matrix} \tag{2}$$

D specifies the distance between the cross-dock stripping and stacking doors. In this problem we have 4 of each. d_{ij} is the distance between strip door i and stack door j. In the present case, $d_{4,1} = 11$, $d_{2,3} = 9$ and so on.

S specifies the capacities on the stripping doors. In this problem we have the matrix/array S:

$$S = 196\ 196\ 196\ 196 \tag{3}$$

The interpretation is that each of the four strip doors has a capacity of 196 units (think: cubic feet of goods) in the planned time window. Similarly, we have

$$R = 196\ 196\ 196\ 196 \tag{4}$$

with the interpretation that each of the stack doors has a capacity of 196 units.

We are now in position to formulate the CDAP as a mathematical program:

$$\text{Minimize} \quad z = \sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{m=1}^{M} \sum_{n=1}^{N} d_{ij} w_{mn} x_{mi} y_{nj} \tag{5}$$

Subject to:

$$\sum_{m=1}^{M} s_m x_{mi} \leq S_i \quad i = 1, 2, \ldots, I \tag{6}$$

$$\sum_{i=1}^{I} x_{mi} = 1 \quad m = 1, 2, \ldots, M \tag{7}$$

$$\sum_{n=1}^{N} r_n y_{nj} \leq R_j \quad j = 1, 2, \ldots, J \tag{8}$$

$$\sum_{j=1}^{J} y_{nj} = 1 \quad n = 1, 2, \ldots, N \tag{9}$$

$$x_{mi} = 0 \text{ or } 1 \quad m = 1, 2, \ldots, M, i = 1, 2, \ldots, I \tag{10}$$

$$y_{nj} = 0 \text{ or } 1 \quad n = 1, 2, \ldots, N, j = 1, 2, \ldots, J \tag{11}$$

The x_{mi}s and y_{nj}s are binary decision variables. x_{mi} has the interpretation 1 if goods of source type m (goods originating at source m) are stripped at strip door i; 0 otherwise. Similarly y_{nj} is 1 if goods of type n are stacked at door j. S_i is the capacity of strip door i. R_j is the capacity of stack door j. These capacities are in cubic feet per shift and so are soft in practice. This is a main reason we seek solutions of interest, as characterized. $s_m = \sum_{n=1}^{N} w_{mn}$ and is the total number of units from source m. Similarly, $r_n = \sum_{m=1}^{M} w_{mn}$ and is the total number of units shipped to destination n.

CDAP is quadratic in the objective, with the x_{mi} and y_{nj} decision variables being multiplied together because the value of assigning a truck to a strip door depends in part on which trucks (with which destinations) are assigned to specific stack doors. The number of binary decision variables under the mathematical programming formulation, above, is $M * I + N * J$. This equals $8 \times 4 + 8 \times 4 = 64$ in the example to hand, the SetA_8x4S30 benchmark CDAP. Experience to date generally finds that exact solvers (branch-and-bound) are unable to solve CDAP problems with 210 binary variables or more, e.g., for problems with 15 sources, 15 destinations, 7 strip doors and 7 stack doors, or larger.

In the next two sections we describe our two solution approaches for solving the models and collecting the SoIs.

3 The FI2PopGA+SoI Algorithm

We used what we call the FI2PopGA+SoI algorithm to solve the CDAP test cases we considered and to collect the SoIs. Our algorithm is a close variant to the "feasible-infeasible two-population GA," described in [11] and [9] (and partially in [10]), augmented by collecting SoIs encountered. Briefly, the FI2Pop GA, consists of two genetic algorithms (GAs, which can be of essentially any kind; crossover and mutation methods are not material to the concept) arranged so one works on a feasible and the other an infeasible population of solutions.

The SoI collection part of the algorithm is intertwined with the FI2PopGA part and works as follows. At initialization two data structures, a dictionary and a priority queue, are set up for each class of SoI to be collected. (We used four in this study; two kinds of feasible and two kinds of infeasible solutions [9].) During evaluation of daughter solutions (from either the feasible or the infeasible population), after each solution is evaluated and its feasibility determined, its presence in the relevant dictionaries is checked. If the solution is present, nothing else is done. If it is not present, then the priority queue associated with the dictionary is checked. If its length is less than the maximum allowed, then the solution is added to both the priority queue and the dictionary. If the length of the priority queue equals the maximum allowed, then the worst element of the queue is popped and its objective value compared to the solution to hand. If the solution to hand is not superior to the popped solution, nothing else is done with that solution and the popped solution is returned to the priority queue. If the solution to hand is superior to the popped solution, then the popped solution is removed from the dictionary, and the solution to hand is pushed to the priority queue and added to the dictionary.

From this description it should be apparent that the added computational cost of collecting the SoIs is quite tractable. Note also that collecting infeasible solutions depends critically on having available a population of them, raising the question of the suitability of solver algorithms without this property.

Regarding prior research we note that [3] treats CDAP with evolutionary computation. Also, Ann Kuo in her thesis [12] successfully treated related difficult test problems with the FI2PopGA. We are not aware of any related efforts to collect SoIs for CDAP with evolutionary computation (or indeed other metaheuristics).

4 The Convex Hull Heuristic (CHH)

The convex hull heuristic (CHH) is a multi-start metaheuristic, designed for pure integer nonlinear differentiable optimization problems with linear constraints. 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. A single start of the algorithm, loosely based on simplicial decomposition [8], alternates between a continuous nonlinear problem (NLP) with one linear constraint, whose number of variables increases by one at each iteration, and one linear integer programming subproblem (IPS) subject to all original constraints. CHH 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.

CHH is based on a relaxation method called convex hull relaxation (CHR), independently introduced in [4] and [2] as an extension of the primal relaxation of [6]. For nonlinear integer minimization problems, CHR computes both a lower bound on the optimal value and good integer feasible solutions. This can be defined for both convex and nonconvex problems, however it only produces a valid bound for convex problems, and it is only efficient as a heuristic in the nonconvex case. In the latter case, the algorithm generates, as a by-product, what are almost always high quality integer feasible solutions.

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 subproblem to optimality. It does happen in practice that out of maybe fifty or sixty calls to the linear MIP solver, one or two problems happen 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), CPLEX 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 objective 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. For a more detailed description of the CH heuristic, see [1, 7].

5 Experiments and Results

In an extended series of studies drawn upon for this paper, we have been working with CDAP test problems generated by Cardoso da Silva [5] for his thesis. In the thesis they were solved by two related local search heuristics. These problems are well accepted in the CDAP research community and are thought to be reasonable approximations of the real world CDAP instances that are known.

Optimal solutions are available, via branch-and-bound solvers, for about 15 of the smaller CDAP test problems. Larger problems defeat branch-and-bound today and so must be approached with heuristics. Given our interest here is on finding solutions of interest, we focus on problems for which optimal solutions are known.

Because of space limitations in this extended abstract, we must severely limit our discussion of results. In particular we report on results for just one (representative) problem, SetA_10x5S5 from Cardoso da Silva [5]. That problem has 5 strip (incoming) doors, 5 stack (outgoing) doors, 10 sources of goods, and 10 destinations.

Tables 1 and 2 show the top feasible and infeasible SoIs discovered by the FI2PopGA+SoI alogrithm. We also solved the SetA_10x5S5 problem using the CHH algorithm, retaining solutions from the CPLEX pool, as described above. The top feasible solutions are reported in Table 3. The top infeasible SoIs are shown in Table 4. Points arising:

1. The CHH executes much faster (13 seconds on a good server) than the FI2PopGA+SoI (a couple of hours for these problems on a good laptop).
2. With reference to Table 3 the CHH found a total of 33 feasible SoIs. CHH also found 32 infeasible SoIs. These are also collected passively (and cheaply).

Table 1. FI2PopGA+SoI: Top feasible SoIs for CDAP SetA_10x5S5. i = solution item number. objval = objective value. ss = sum of slacks. sN = slack on constraint N.

i	objval	ss	s0	s1	s2	s3	s4	s5	s6	s7	s8	s9
0	6616	78	0	0	9	5	25	6	0	12	7	14
1	6618	78	0	0	16	5	18	6	0	12	7	14
2	6622	78	0	10	6	5	18	6	0	12	7	14
3	6631	78	0	0	16	5	18	6	0	7	12	14
4	6635	78	0	10	6	5	18	6	0	7	12	14
5	6651	78	0	3	6	5	25	6	0	12	7	14
6	6661	78	0	0	9	5	25	6	0	7	12	14
7	6667	78	0	10	18	5	6	6	14	7	0	12
8	6670	78	0	0	9	5	25	0	6	12	7	14
9	6671	78	0	25	3	5	6	6	7	14	0	12

Table 2. FI2PopGA+SoI: Top infeasible SoIs for CDAP SetA_10x5S5. i = solution item number. objval = objective value. sN = slack on constraint N.

i	objval	s0	s1	s2	s3	s4	s5	s6	s7	s8	s9
0	6417	36	0	-16	-6	25	6	0	12	7	14
1	6418	36	0	-16	1	18	6	0	12	7	14
2	6440	36	0	-20	5	18	6	16	6	-3	14
3	6460	-6	16	6	5	18	6	-10	32	-3	14
4	6463	36	0	-20	5	18	6	39	-17	-3	14
5	6464	36	0	-20	5	18	6	0	22	-3	14

Some come from the pool of interim solutions found by the CPLEX solver; others come directly from the simplicial decomposition steps. In consequence, the number of SoIs (feasible or infeasible) to be collected is really not under control of the user, and with the process being multi-start, the same solution can be found in several different passes. The FI2PopGA+SoI on the other hand found hundreds of distinct SoIs, both feasible and infeasible, and the number to collect is a parameter in the algorithm.

3. It is interesting to compare the (extended) tables from the two methods. The overlap in SoIs discovered is small, suggesting that employing both methods (and in general multiple methods) would be useful for discovering SoIs.

6 Discussion and Conclusion

We have done an extensive number of runs with different CDAP test cases, including at the high end a 100 source and destination problem, with 30 strip and 30 stack doors. This is larger than can be handled by contemporary branch-and-bound solvers, yet our two heuristics were successful in finding good solutions in the sense articulated in the first paragraph of this paper. Space limitations prevent their presentation here, but what we have reported is representative.

Several points emerge. First, we have demonstrated success in collecting both feasible and infeasible SoIs with CDAPs, using our FI2PopGA + SoI algorithm

Table 3. CHH: Top feasible SoIs for CDAP SetA_10x5S5. i = solution item number. objval = objective value. sN = slack on constraint N.

i	objval	s0	s1	s2	s3	s4	s5	s6	s7	s8	s9
0	6616	25	5	9	0	0	14	7	12	0	6
1	6618	18	5	16	0	0	14	7	12	0	6
2	6640	0	10	18	5	6	6	0	14	7	12
3	6640	0	10	18	5	6	6	0	14	7	12
4	6640	0	10	18	5	6	6	0	14	7	12
5	6640	6	5	18	10	0	12	7	14	0	6
6	6640	6	5	18	10	0	12	7	14	0	6

Table 4. CHH: Top infeasible SoIs for CDAP SetA_10x5S5. i = solution item number. objval = objective value. sN = slack on constraint N.

i	objval	s0	s1	s2	s3	s4	s5	s6	s7	s8	s9
0	6361	18	-15	-16	16	36	14	-6	35	-10	6
1	6387	-12	15	-16	16	36	14	-6	35	-10	6
2	6410	-12	15	-16	16	36	14	17	12	-10	6
3	6418	36	-16	16	-15	18	6	35	-10	-6	14

as well as our CHH algorithm. The SoIs discovered, or at least some of them, are plausibly useful in decision making for sensitivity analysis and other post-solution tasks. Moreover, all of this is quite general and not dependent upon CDAPs per se. (We have also investigated GAP in some depth.) Second, with evolutionary computation, specifically with our FI2PopGA + SoI algorithm, we find it productive to lengthen the runs to a point well beyond what is normally needed to find an optimal solution. Populating the SoIs is itself a challenging search problem. Third, the CHH is very much faster than FI2PopGA + SoI algorithm. In the present case, the CHH finds fewer SoIs, but this can be changed in principle by keeping more solutions from the CPLEX solution pool. This raises the prospect of systematically varying the constraint RHS values and running the CHH multiple times, thereby undertaking a parameter sweep on the constraints. This is indeed possible, and may often be advisable, but it does encounter scaling problems. Sweeping through 10 RHS values for a problem with 12 constraints presents 10^{12} problems to be solved. No doubt, however, much of value can be obtained from a judicious, partial sweeping with the CHH.

Much, of course, remains to be learned. On CDAP, actual case studies need to be done and practical evaluation of SoIs undertaken. More generally, we envision these ideas being taken up systematically so that instead of only an optimal solution for a given problem, decision makers are presented with a rich body of SoI data, which they can use for deliberation and which, with automated support, can be used to suggest opportunities. To this end, new concepts need to be conceived (e.g., for defining SoIs and making use of them), and DSS techniques will need to be developed. Beyond that, we believe these results open up the prospect of a rich new stream of research for addressing the question of how best to populate the SoIs. Metaheuristics will surely be an important part of the answer, as will variations on the CHH and other mathematical programming metaheuristics that can produce multiple solutions. Also, algorithms for CSPs (constraint satisfaction problems) and related problems such as MaxCSP and MaxSAT, often produce multiple solutions or partial solutions and so are worth investigating for their potential to populate the SoIs. Finally, all of the metaheuristics used for constrained optimization (vast in number) need to be explored for how they may contribute to populating SoI sets and complement other methods. The amount of useful work to be done to explore generalized optimization thoroughly is indeed immense.[1]

References

1. Ahlatçioglu, A., Bussieck, M., Esen, M., Guignard, M., Jagla, J., Meeraus, A.: Combining QCR and CHR for convex quadratic pure 0-1 programming problems with linear constraints. Annals of Operations Research 199(1), 33–49 (October 2012)
2. Ahlatçioglu, A., Guignard, M.: The convex hull relaxation for nonlinear integer programs with linear constraints. Technical report, University of Pennsylvania, The Wharton School, OPIM Department, Philadelphia, PA (2007/2009), latest revision January 2009
3. Aickelin, U., Adewunmi, A.: Simulation optimization of the crossdock door assignment problem. CoRR abs/0803.1576 (2008), `http://arxiv.org/abs/0803.1576`
4. Albornoz, V.: Diseno de Modelos y Algoritmos de Optimizacion Robusta y su Aplicacion a la Planificacion Agregada de la Produccion. Ph.D. thesis, Universidad Catolica de Chile, Santiago, Chile (1998)
5. Cardoso da Silva, D.: Uma heuristica hibrida de busca em vizinhança larga para o problema de atribuiçño de portas de cross-dock. Ph.D. thesis, Universidade Federal fluminense, Escola de Engenharia (2013), mestrado em Engenharia de Produção
6. Guignard, M.: Primal relaxations for integer programming. Invited plenary session, VII CLAIO, Santiago, Chile (July 1994)
7. Guignard, M., Hahn, P.M., Pessoa, A.A., Cardoso da Silva, D.: Algorithms for the cross-dock door assignment problem. In: Proceedings of the 4th International Workshop on Model-Based Metaheuristics. pp. 1–12. Angra dos Reis, Brazil (2012)
8. Hohenbalken, B.V.: Simplicial decomposition in nonlinear programming algorithms. Mathematical Programming 13, 49–68 (1977)
9. Kimbrough, S.O., Lau, H.C.: Business Analytics for Decision Making. CRC Press, Boca Ratan, FL (2016)
10. Kimbrough, S.O., Lu, M., Wood, D.H., Wu, D.J.: Exploring a two-population genetic algorithm. In: Cantú-Paz, E., et al. (eds.) Genetic and Evolutionary Computation (GECCO 2003). pp. 1148–1159. LNCS 2723, Springer, Berlin, Germany (2003)
11. Kimbrough, S.O., Koehler, G.J., Lu, M., Wood, D.H.: On a feasible–infeasible two–population (FI-2Pop) genetic algorithm for constrained optimization: Distance tracing and no free lunch. European Journal of Operational Research 190(2), 310–327 (2008)
12. Kuo, A.: Unveiling Hidden Values of Optimization Models with Metaheuristic Approach. Ph.D. thesis, University of Pennsylvania, Philadelphia, PA (May 2014)

[1] An extended working paper is available from the authors.