Chapter 3

Column Generation Approach for the Location-Routing Problem

Introduction

In this chapter we present an alternative model for the LRP described in the previous one. The variables in this model are associated with feasible routes rooted at each of the potential sites. The number of such variables suggests the use of Column Generation (CG), which has been previously applied to different VRPs (Desrochers, Desrosiers, and Solomon 1992; Dror 1994; Desrosiers, Dumas, Solomon, and Soumis 1995). In our case, the convexification of the master problem is obtained via Lagrangian relaxation. A similar approach was used in Díaz (2001) and Díaz and Fernández (2002) for the SSCPLP.

The pricing problem that has to be solved in our CG approach to identify new columns is rather involved. It consists of finding tours starting and ending at the same plant, with associated negative cost and that do not violate the capacity constraint of the plant. We show that, in fact, it can be seen either as an ESPPRC (Gueguen, Dejax, Dror, Feillet, and Gendreau 2000) or as a *Knapsack Constraint Profitable Tour Problem* (KCPTP) (Gensch 1978; Göthe-Lundgren, Maffioli, and Värbrand 1995), that is a special case of the TSP with profits (Feillet, Dejax, and Gendreau 2001), depending on whether we use or not the auxiliary network presented in the previous chapter. In our case, we have regarded it as an ESPPRC, which was proven to be $\mathcal{NP}$-hard in the strong sense, in Dror (1994). In this chapter, we present a simple heuristic that allows us to avoid the exact solution of the ESPPRC in many occasions. When this is not possible, the code proposed by Gueguen, Dejax, Dror, Feillet, and Gendreau is used.

The structure of this chapter is as follows: In Section 3.1 we present the proposed model. In Section 3.2 we derive the subproblems to solve within the CG scheme; the master problem and the pricing subproblem. We also discuss how we actually generate the new columns and we outline the algorithm presented in Gueguen, Dejax, Dror, Feillet, and Gendreau (2000) for the ESPPRC. Then, we present the structure of the whole algorithm in Section 3.3. We conclude this chapter with some computational experiences and final comments in Sections 3.4 and 3.5, respectively.

3.1 Formulation

We recall the notation used in previous sections:

I set of potential plant locations

J set of customers

b_i capacity of plant i, $i \in I$

f_i fixed costs for opening plant i, $i \in I$

d_j demand of customer j, $j \in J$

c_{rs} arc costs, for $r, s \in I \cup J$, not both in I.

For each potential plant location $i \in I$ we define K_i as the index set of all routes leaving plant i, and satisfying a total demand that does not exceed the capacity b_i. Let $k \in K_i$ be the index of such a route. Given a customer j, column coefficient a_j^k takes the value 1 if route k visits plant j and 0 otherwise.

If, for a route $k \in K_i$, we denote

$$J_k = \{j \mid a_j^k = 1\},$$

the cost associated with route k can be expressed as

$$p_k = f_i + TSP\Big(\{i\} \cup J_k\Big), \tag{3.1}$$

where $TSP(S)$ is the optimal value of the TSP defined on nodes in S for $S \subseteq I \cup J$.

Using this notation, the LRP can be stated as:

$$(LRP) \quad \text{minimize} \quad \sum_{i \in I} \sum_{k \in K_i} p_k z_k \tag{3.2}$$

$$\text{subj. to} \quad \sum_{k \in K_i} z_k \leqslant 1 \qquad i \in I \tag{3.3}$$

$$\sum_{i \in I} \sum_{k \in K_i} a_j^k z_k = 1 \quad j \in J \tag{3.4}$$

$$z_k \in \{0, 1\} \qquad i \in I, k \in K_i \tag{3.5}$$

Constraints (3.3) forbid solutions where more than one route is selected for a fixed plant location, while constraints (3.4) ensure that each customer is visited by exactly one of the selected routes. These constraints can be seen as assignment constraints or partitioning cosntraints.

If we relax the assignment constraints in a Lagrangian fashion we obtain:

$$LRPL(u) = \sum_{j \in J} u_j + \text{ minimize } \sum_{i \in I} \sum_{k \in K_i} \left(p_k - \sum_{j \in J} u_j a_j^k \right) z_k \tag{3.6}$$

$$\text{subj. to } \sum_{k \in K_i} z_k \leqslant 1 \qquad i \in I \tag{3.3}$$

$$z_k \in \{0,1\} \qquad i \in I, k \in K_i \tag{3.5}$$

And the Lagrangian dual:

$$(LRPLD) \quad \text{maximize} \quad LRPL(u)$$
$$\text{subj. to} \quad u_j \in \mathbb{R} \qquad j \in J$$

Given a vector u, $LRPL(u)$ is separable in the set of plants, and each subproblem can be easily solved as follows.

For each potential plant location $i \in I$, let

$$dc_i(u) = \min_{k \in K_i} \left\{ p_k - \sum_{j \in J} u_j a_j^k \right\} \tag{3.7}$$

and $k(i,u)$ such that

$$dc_i(u) = p_{k(i,u)} - \sum_{j \in J} u_j a_j^{k(i,u)}, \tag{3.8}$$

the optimal solution to $LRPL(u)$ is given by:

$$z_k^* = \begin{cases} 1 & \text{if } k = k(i,u) \text{ and } dc_i(u) \leqslant 0 \\ 0 & \text{otherwise.} \end{cases}, \quad k \in K_i, i \in I.$$

The linear relaxation $LRPL(u)$ can be reinforced by adding the following constraint:

$$\sum_{i \in I} b_i \sum_{k \in K_i} z_k \geqslant D, \tag{3.9}$$

where D is the aggregated demand, or a lower bound on the total capacity of the optimal set of plants. This constraint forces the set of open plants to have enough capacity as to service the aggregated demand. This is not guaranteed otherwise since the constraints that ensure that there is enough overall capacity are the assignment constrants that have been relaxed.

If constraint (3.9) is added to $LRPR(u)$, its solution reduces to computing $dc_i(u)$ for each $i \in I$, and then solving the knapsack problem:

$$KP(u) = \text{ minimize } \sum_{i \in I} dc_i(u) s_i$$
$$\text{subj. to } \sum_{i \in I} b_i s_i \geqslant D$$
$$s_i \in \{0,1\} \qquad i \in I,$$

to decide the set of plants to open. Their associated routes will be then given by the respective values of $k(i,u)$.

3.2 The Pricing Problem

As described in Section 3.1, each column in LRP corresponds to a feasible route for one of the potential plants; the elements in the column determine which customers are serviced by the route, and its cost is the cost for opening the plant, plus the cost of the an optimal route visiting that set of customers.

Given a dual vector u, the price in LRPLD of a column z_k ($k \in K_i$) is given by

$$\tilde{p}_k = p_k - \sum_{j \in J} u_j a_j^k. \tag{3.10}$$

Finding columns that have negative price for a given plant is equivalent to finding tours rooted in the plant, such that the capacity constraint is satisfied and the sum of the multipliers associated with each of the visited customers is higher than the cost of opening the plant plus the route cost. The problem of finding the feasible route with smallest price is known as the KCPTP.

Alternatively, we can consider the auxiliary graph $G_i = (V_i, A_i)$, where $V_i = J \cup \{i, i'\}$, and

$$A_i = \{(i,j)|\ j \in J\} \cup \{(j_1,j_2)|\ j_1, j_2 \in J; j_1 \neq j_2\} \cup \{(j,i')|\ j \in J\},$$

with the following cost function defined on the set of arcs:

$$\hat{c}_e = \begin{cases} c_e & \text{if } e = (j,i'),\ j \in J \\ c_e - u_j & \text{if } e = (s,j),\ s \in I \cup J, J \in J. \end{cases}$$

Note that $G_i(V_i, A_i)$ is the subgraph of the graph G presented in Chapter 2, defined by the plant i and the set of customers (see Figure 2.1). Using the equivalency between tours and paths in G_i explained in the previous chapter, we can represent any valid column by an ii'–path in G_i. The cost of this path coincides with the price of the column. So, the KCPTP problem stated above is equivalent to finding an elementary shortest ii'–path in G_i that satisfies the capacity constraint on plant i. This is known as the ESPPRC, already studied in the literature, also within the context of column generation for VRPs. In this work we have chosen this second approach.

As menctioned above, the ESPPRC is $\mathcal{NP}$-hard. So, its exact resolution requires a considerable computational effort, in general. However, since any valid column with negative price is a candidate to improve the current solution of LRPLD, regardless of being or not the optimal solution to the ESPPRC, a heuristic method can be used instead of an exact algorithm. Our proposal is to initially

try to solve the pricing problem with a heuristic that looks for columns (associated with each potential plant location) that price properly. Alternatively, the heuristic can prove that no such column exists for any plant location. We proceed as follows:

With each customer j we associate the profit

$$v_j = u_j - \underline{c}_j,$$

where $\underline{c}_j$ is the distance from customer j to its closest point, either another customer or a potential plant location. Then, we try to find the set of customers with the highest total profit that satisfies the capacity constraint, by solving the following knapsack problem:

$$\begin{aligned} KP_i(u) = \max & \sum_{j \in J} v_j w_j \\ & \sum_{j \in J} d_j w_j \leqslant b_i \\ & w_j \in \{0,1\} \qquad j \in J \end{aligned}$$

Proposition 3.2.1. *If $KP_i(u) \leqslant f_i$, then no column with negative price can be found for plant location i.*

Proof.

This proposition follows from the fact that $f_i - KP_i(u)$ is a lower bound on the smallest price of a valid column associated with plant location i:

Let $k \in K_i$, and J_k its associated set of customers. Then, by equations 3.1 and 3.10,

$$\tilde{p}_k = f_i + TSP\Big(J_k \cup \{i\}\Big) - \sum_{j \in J_k} u_j$$

Now, since all the distances are positive,

$$TSP\Big(J_k \cup \{i\}\Big) \geqslant \sum_{j \in J_k} \underline{c}_j,$$

and we obtain

$$\tilde{p}_k \geqslant f_i + \sum_{j \in J_k} \underline{c}_j - \sum_{j \in J_k} u_j.$$

Grouping the two last terms we can deduce:

$$\tilde{p}_k \geqslant f_i - \sum_{j \in J_i} (u_j - \underline{c}_j) \geqslant f_i - KP_i(u)$$

□

Proposition 3.2.2. *If $KP_i(u) > f_i$, let w^* be an optimal solution with this value and*

$$J_i = \{j \in j \,|\, w_j^* = 1\} \quad \text{and} \quad p_{k'} = f_i + TSP\Big(J_i \cup \{i\}\Big) - \sum_{j \in J_i} u_j.$$

Then, the column with entries $a_j^{k'} = w_j^*$, is valid for plant location i and has price $p_{k'}$.

Proof. This result follows from equation (3.10). □

Note that the price of this new column is likely to be small, since the path goes through customers with high profit.

Only when the heuristic fails, the algorithm by Gueguen et al. is applied. We try to avoid using this algorithm since it is quite time consuming, specially in problems where the capacity constraint is not tight. This algorithm is a multiple labelling approach, where feasible labels are only fathomed when dominated. Labels include the cost of partial solutions, the consumed resource and the already visited customers. Cycling is avoided by considering each customer as a resource whose capacity is 1.

3.3 Algorithm Structure

The structure of the proposed approach is presented in Algorithm 3.1. It consists of successively solving the Lagrangian dual LRPLD restricted to the columns at hand, using subgradient optimization, and, once the optimal multipliers have been identified, seeking new columns with negative price with respect to those multipliers. The algorithm terminates when no such column exists.

Algorithm 3.1 LRP_CG

Let $\bar{z}$ be an initial solution and $O(\bar{z})$ its set of open plants
Initialize $K_i, i \in O(\bar{z})$ with the column associated with the customers assigned to plant i in $\bar{z}$. [a]
Set ub to the value of $\bar{z}$.
$StopCriterion \leftarrow$ **false**
while (**not** $StopCriterion$) **do**
  Solve_LRPLD$(K, ub) \longrightarrow u^*, z(u^*), lb$.
  if ($z(u^*)$ defines a feasible solution and $lb < ub$) **then**
    $ub \leftarrow lb$
    $\bar{z} \leftarrow z(u^*)$
  end if
  $new \leftarrow$ Column_Generate(u^*)
  if (new = **true**) **then**
    Update K
  else
    $StopCriterion \leftarrow$ **true**
  end if
end while
lb is a lower bound for the LRP.
if ($ub = lb$) **then**
  $\bar{z}$ is the optimal solution.
end if

[a] The optimal route through these customers must be found to assign the column the appropriate cost.

In Algorithm 3.1 calls are made to two other algorithms; Solve_LRPLD(K, ub) and Column_Generate(u^*). Solve_LRPLD applies subgradient optimization to solve the Lagrangian Dual LRPLD restricted to the

columns in K, as outlined in Algorithm 3.2. Observe that

$$\delta_j = 1 - \sum_{i \in I} \sum_{k \in K_i} z_k a_j^k, \quad j \in J,$$

gives an element of the subgradient of $LRPL(u)$.

On the other hand, Column_Generate (Algorithm 3.3) either finds columns with negative price with respect to the current dual multipliers or proves that all columns have non negative price.

Algorithm 3.2 Solve_LRPLD(K, ub)

```
Initialization: u ← u_0, StopCriterion ← false
while (not StopCriterion) do
  Compute dc_i(u) and k(i,u) for each i ∈ I as defined in (3.7)-(3.8).
  Solve kp(u) ⟶ z, LRPL(u)
  Compute the subgradient ⟶ δ
  if ( ||δ|| ⩽ ε or ||LRPL(u) − ub|| ⩽ ε ) then
    StopCriterion ← true
  end if
  if (not StopCriterion) then
    Update u
  end if
end while
return (u, z, LRPL(u))
```

This process for solving the pricing problem is sketched in Algorithm 3.3. We use ESPPRC_solve(i, u) to denote calls to the exact algorithm to solve the problem for plant i, with dual variable values u.

3.4 Computational Results

We applied the proposed algorithm to the smallest problems of the test set presented in the former chapter, *i.e.*, to those in the group S1, that have 5 plants and 10 customers. At each run, the program was interrupted if it had not terminated after 2 hours of CPU time. The obtained results are presented in Table 3.1. In the initialization step, the solution obtained with the Tabu Search proposed in Chapter 2 is used.

All computations have been carried out on a SUN Ultra-10/333 workstation, SPECint95 14.1. The code used to solve the different KPs has been taken from Martello and Toth (1990), while we have used the algorithm presented in Toth and Carpaneto (1995) for the ATSP.

The first column of the table indicates de name of the instance. In the next two columns we find the values of the upper and the lower bounds obtained with Algorithm 3.1. CPU times (in seconds) are shown in the fourth column, while the fifth column contains the number of iterations performed. The last two columns refer to the number of times that the exact algorithm for the ESPPRC had to be used. Column 6 depicts the number of iterations where the heuristic did not provide suitable information, and column 7 gives the actual number of calls to ESPPRC_solve. Note that the number of calls to the exact solver for the ESPPRC can vary from one iteration to another, because it is called for each plant until a column with negative price is found or all the plants have been checked.

Even for the small size considered, there were some instances where the algorithm could not finish within the limit of 2 hours of CPU time. In those instances, the program was stalled in one of the calls

Algorithm 3.3 Column_Generate(u)

Initialization: $ColFound \leftarrow$ **false**; $bestcols \leftarrow (0, \ldots, 0)$

for $(i \in I)$ **do**
  Solve $KP_i(u) \longrightarrow J_i$
  if $(KP_i(u) \leqslant f_i)$ **then**
    {No improving column exists for this plant}
    $bestcols[i] \leftarrow \infty$
  else
    Solve $TSP\Big(J_i \cup \{i\}\Big)$ to find $p_{k'}$
    $bestcols[i] \leftarrow p_{k'}$
    if $(p_{k'} < 0)$ **then**
      {Column with negative price}
      $ColFound \leftarrow$ **true**
    end if
  end if
end for

{All plants checked; resort to ESPPRC_solve if necessary}
for $(i \in I)$ **do**
  if ((**not** $ColFound$) **and** $(bestcols[i] < \infty)$) **then**
    $ColFound \leftarrow$ ESPPRC_solve(i, u)
  end if
end for

return $ColFound$

to ESPPRC_solve. For the other instances, however, the results obtained are very encouraging, since the gap between the upper and the lower bound is always 0, and the computational times are small. Unfortunately, for the instances where a call to ESPPRC_solve does not terminate in a reasonable amount of time, it is not possible to validate the bound at hand. So, in those cases, the algorithm cannot provide any lower bound for the LRP.

3.5 Conclusions

In this chapter we propose a CG scheme to find lower bounds for the deterministic LRP under consideration in this work. As it is the case in most applications of CG to VRPs, the backbone of the algorithm is the resolution of the pricing problem, that in our case can be seen either as a PCTSP or as an ESPPRC. In our approach we have interpreted it as an ESPPRC, and we have designed a simple heuristic to avoid solving it in the majority of the cases. However, its exact resolution is required in some occasions to validate the bound obtained with the algorithm. In those cases, we have used the code presented in Gueguen, Dejax, Dror, Feillet, and Gendreau (2000). Even if it is an implementation of a recent algorithm for the ESPPRC and it is quite efficient in general, in some cases it has failed to provide us with the requested solutions within reasonable amounts of time, because of the looseness of the capacity constraints.

Due to the good quality of the bounds obtained when the algorithm succeeds to terminate within the pre-specified time limit, we consider that the improvement of the existing exact algorithms for the

Table 3.1: Results of Column Generation for Problems in Group S1

PROB.	CG	UPPER	TIME	iter	e_iter	e_calls
1S1a	1159.17	1159.17	58.24	4	2	8
2S1a	1296.78	1296.78	234.99	5	5	18
3S1a	1139.71	1139.71	74.38	2	2	7
4S1a	1174.93	1174.93	118.70	5	5	14
5S1a	1119.94	1119.94	112.81	4	4	13
1S1b	1159.17	1159.17	82.77	5	2	8
2S1b						
3S1b	1139.71	1139.71	188.75	5	5	17
4S1b	1174.93	1174.93	82.59	3	3	10
5S1b	1333.08	1333.08	38.20	2	1	5
1S1c	2208.80	2208.80	21.46	16	3	3
2S1c	2165.50	2165.50	68.39	13	4	8
3S1c						
4S1c	1974.18	1974.18	30.04	15	2	7
1S1d	2381.21	2381.21	15.23	13	1	1
2S1d						
3S1s	1199.30	1199.30	7.14	24	1	5
4S1d						
5S1d	2306.77	2306.77	35.36	21	4	4
1S1e	3433.68	3433.69	74.04	25	14	26
2S1e						
3S1e	2987.49	2987.49	19.04	16	4	11
4S1e	3197.78	3197.78	4.00	10	1	1
5S1e						

ESPPRC or the development of new ones, specialized in problems with loose capacities, constitute a promising future research venue.

The work presented in this chapter suggests two direct extensions. On the one hand, an exact algorithm for the LRP can be obtained by including this approach in a branch and price scheme and, in the other hand, the generalization of this approach to the case with multiple vehicles is straight forward.