

Planning Problems in Arc Routing

Philippe Lacomme

LIMOS, Université Blaise Pascal, Clermont-Ferrand, France.
lacomme@sp.isima.fr

Christian Prins (corresponding author) **and Wahiba Ramdane-Chérif**

LOSI, Université de Technologie de Troyes, France.
prins@utt.fr, ramdane@utt.fr

Abstract

Planning problems over multi-period horizons occur in arc routing but are neglected by OR researchers. This paper presents these new problems, a classification and a hybrid GA able to tackle the planning and scheduling levels simultaneously.

1. Introduction to arc routing problems

Arc routing problems are raised by applications in which a subset of arcs in a network must be processed at a minimum total cost, *e.g.* urban waste collection or inspection of power lines. Although our research tackles general problems (mixed networks, prohibited turns, etc.), we restrict here the presentation to the basic undirected problem, due to paper length constraints.

The basic *Capacitated Arc Routing Problem* (CARP) considers an undirected graph $G = (V, E)$ with a set V of n nodes and a set E of m edges. In street networks, an edge corresponds to a bi-directional street whose both sides can be processed during the same traversal and in any direction. A fleet of nv identical vehicles of capacity Q is available at a depot node s . Each edge $(i, j) = (j, i)$ has a traversal cost $c_{ij} \geq 0$, a demand $r_{ij} \geq 0$ and can be traversed any number of times. A subset R of nt edges, the tasks, must be serviced by the fleet. All data are integers. The CARP consists in computing a set of vehicle trips of minimum total cost, such that each trip starts and ends at the depot, each task is serviced by one single trip, and the load of each vehicle never exceeds Q .

Lacomme, Prins and Ramdane-Chérif have published in [2] an efficient genetic algorithm (GA) for the CARP. On two sets of classical benchmarks, it outperforms the best metaheuristics published and reaches for 70% of instances a tight lower bound proposed by Belenguer and Benavent [1]. A natural continuation was to extend this GA to multi-period problems.

2. Planning problems in arc routing

Consider a horizon H of np periods or "days", in which each task (i, j) must be serviced f_{ij} times, $1 \leq f_{ij} \leq np$, but at most once in each day. The total number of services ns is then the sum of all task frequencies. The *arc routing planning problem* or *periodic CARP* (PCARP) consists in assigning each task to f_{ij} distinct days to minimize the total cost of the trips over H . The PCARP is obviously NP-hard since it includes the basic (single-period) CARP. It remains NP-hard even for $np = nv = 2$, $c_{ij} = 0$ for all edges, $f_{ij} = 1$ for all tasks, and a total demand of $2 \cdot Q$: the residual problem is still a NP-hard *set partitioning problem*.

Compared to production management, the assignment phase corresponds to a production planning problem (*tactical* decision level), while the detailed solution of the CARP for one day corresponds to a scheduling problem (*operational* level). In production planning, schedules are often computed for the first period of H only, because other periods are still too uncertain. In most arc routing problems like waste collection, this *rolling horizon* method is not valid because service days for each street must be known in advance by the population and remain stable for typically one year. In this context, *it is possible to solve simultaneously the planning and scheduling levels*, since each scheduling problem (daily CARP) is entirely defined after the assignment.

3. Complications

3.1. Varying demands - A simple classification

When each cost is a duration, it usually grows with the demand. The basic CARP is not realistic with its unique cost c_{ij} per edge traversal, whether this edge is serviced or not. For the PCARP, we keep c_{ij} for a deadheading traversal but introduce a separate service cost w_{ij} . This distinction was already necessary in [2] for CARP extensions like a maximum trip duration.

In general, service costs can be computed when demands are known. We group into a *class A* the simplest problems in which the demands do not depend on a "production" which accumulates with time, *i.e.* the demand of each edge (and its service cost) will be the same for any service day. Examples are inspection of power lines, pulverization of herbicides to avoid weeds, etc.

The *class B* gathers problems like municipal waste collection in which each task (i,j) has now a production $prod(i,j,p)$ for each day p . Two particular cases are a constant daily production $prod(i,j,p) = prod(i,j)$ or a global trend on the network, *e.g.* peak of waste from lawn mowers after a hot and rainy period. Global trends can be modelled with a reference daily production $ref(i,j)$ for task (i,j) and a growth factor $\alpha(p)$: the production for day p is then $prod(i,j,p) = ref(i,j) \cdot \alpha(p)$.

Class B problems may be *acyclic* (*subclass B1*) or *cyclic* (*subclass B2*). In B1, the horizon is not cyclic and the demand r_{ijp} of (i,j) in day p is the production accumulated since the last service day q (if p is the first service, we assume a *fictitious service* at time 0). An example is when a new road shoulder mowing campaign starts in spring : the vegetation stops growing in winter and the planning of last year can be forgotten. The demand r_{ijp} to be satisfied at day p is then:

$$r_{ijp} = \sum_{k=q+1}^p prod(i, j, k)$$

For cyclic problems B2, H tiles a longer horizon H' , and task-to-day assignments must be identical for any occurrence of H . In waste collection for instance, the amount collected for the first service in a week depends on the waste produced since the last visit in the previous week. Problems B2 are difficult to tackle by dynamic programming algorithms or sequential heuristics that compute PCARP solutions day by day: for instance, the trips for Monday cannot be computed as long as the assignment is not specified for all days (demands are still undefined).

3.2. Spacing constraints

Spacing constraints may be fixed at the strategic level and included in the data. For a task (i,j) , they are defined either by minimum and maximum time lags $spmin(i,j)$ and $spmax(i,j)$ between two services, or by a set $comb(i,j)$ of allowed *day combinations*, *e.g.* $(1,2,\dots,np)$ for "everyday" or $(1,3)$ for "Monday-Wednesday". In waste collection, all combinations in $comb(i,j)$ contain f_{ij} days because frequencies are computed before, according to the "productivity" of each district.

After several attempts to design a GA an LP models, we found time lags more difficult to handle and selected the day combinations, in which spacing is implicitly satisfied. This system is not restrictive because any pair $(spmin(i,j), spmax(i,j))$ can be converted into day combinations. Moreover, the conversion is not costly for small horizons like in waste collection. In Troyes for instance, $np = 7$, the streets are collected everyday (town center) or twice a week (suburbs), but never on the week-ends, and only three combinations are permitted: $(1,2,3,4,5)$, $(1,4)$ and $(2,5)$.

3.3. Fleet size constraint - Relaxation in objective functions

When only nv vehicles are available, an assignment of tasks to days is feasible if the tasks in each day can be partitioned into at most nv vehicle loads not greater than Q . This problem is already NP-complete since it boils down to a bin-packing problem for $np = 1$. In constrained cases, our first crossovers failed to build feasible children, even with a dedicated repairing procedure. We

then designed an integer linear program for the assignment phase that cannot be detailed here. Unfortunately, it is too time-consuming to be used at each crossover of a GA.

In fact, in response to a request for proposals issued by a city, waste management companies try to estimate the minimal resource investment (fleet size, main objective) and then the minimal operating costs (total cost of trips possible with that fleet, secondary objective). We then decided to *relax* the fleet size constraint and to include it in the objective function.

Let S a solution costing $Cost$ and using nvu vehicles, and M a constant always larger than $Cost$. Our GA accepts solutions with more than nv trips per day, but progressively gets rid of them by minimizing the

4.3. Reproduction step

The *incremental replacement* is used. At each iteration, two parents P1 and P2 are randomly selected by a *tournament method*. A generalization of the classical LOX (for acyclic PCARPs) or OX (for cyclic PCARPs) crossovers is then applied to get two children C1 and C2. One child is discarded at random. If no duplicate cost occurs, the other replaces in *Pop* a chromosome drawn above the median cost. For instance, the generalized LOX is given below for child C1 (C2 is obtained by permuting P1 and P2). $day(S,k)$ is the day containing a task occurrence $S(k)$ in a chromosome S and $numb(i,j)$ is the number of occurrences already copied into the child.

```
prepare np empty sublists CL(1),CL(2),...,CL(np) for C1 (one per day)
for each task (i,j), initialize numb(i,j) to 0
draw randomly two cutting sites a and b in P1
for k := a to b //copy P1(a)..P1(b) into C1 (keep sequence and days)
    (i,j) := P1(k); p := day(P1,k)
    copy task (i,j) at the end of CL(p) and increment numb(i,j)
endfor
for k := 1 to ns //scan P2 to finish C1
    (i,j) := P2(k)
    if numb(i,j) < f(i,j) then //if frequency not yet satisfied
        if (i,j) can keep its day in C1 according to comb(i,j) then
            p := day(P2,k)
        else //day must be changed, another day is always possible
            p := 1st day for (i,j) in C1 compatible with comb(i,j)
        endif
        copy (i,j) at the end of CL(p) and increment numb(i,j)
    endif
endfor.
```

5. Concluding remarks

The other GA parameters like stopping criteria and mutation are not yet definitely fixed, several versions are being tested. Due to the lack of PCARP instances in the literature, we have produced our own benchmarks by adding a one-week horizon and day combinations in classical benchmarks for the basic CARP. We compare three solution methods. The 1st one builds a PCARP solution in two phases by solving the tactical problem (task-to-day assignment) and then running a constructive heuristic for each operational problem (daily CARP). The 2nd one is analogous, except that the GA of [2] is used to get an excellent solution for each CARP. The 3rd method tackles the tactical and operational levels simultaneously, using the extended GA of section 4.

The testing is not finished and more details will be given at the conference. On the few instances tested, on average, method 2 improves the solutions of method 1 by 5%, and we already have a version of method 3 that saves 5% more. As said in introduction, our algorithms are not restricted to the undirected PCARP: they can tackle extensions like forbidden turns, mixed graphs, vehicle ranges, etc. These promising results show the interest of integrating tactical and operational decisions in long-term planning problems raised by many applications of arc routing.

References

- [1] Belenguer, J.M. and Benavent, E. (1997). *A cutting plane algorithm for the capacitated arc routing problem*, Research Report, Dept. of Statistics and OR, University of Valencia, Spain.
- [2] Lacomme, P., Prins, C. and Ramdane-Chérif, W. (2001). A genetic algorithm for the capacitated arc routing problem and its extensions, in *Applications of evolutionary computing*, E.J.W Boers (ed.), Lecture Notes in Computer Science 2037, Springer.