CLP SOLUTION FOR NEW VARIANT OF VRP

Rafał SZKLARCZYK

e-lan s.c.
Piłsudskiego Street 42/2, 43-300 Bielsko-Biała, Poland, e-mail: rafal.szklarczyk@e-lan.pl

Summary

In the paper a mathematical model of MDMGVRP is presented. The two CLP programs solving above are described and discussed. The new variant of VRP consider certain number of commodities (goods) transported. In classic VRP there is only one commodity. The second main difference to classic problem is limited stock in depots. This two aspects cause the problem more complicated and prevent from direct applying of common known algorithms for solving the MDMGVRP. Two CLP solver tools were used to formulate CLP programs. The first was Cosytec CHIP the other was GNU-Prolog. Finally a demonstration problem is presented and solution is discussed.

Keywords: vehicle routing problem, VRP, MDMGVRP, CLP, CHIP, GNU-Prolog.

ROZWIĄZANIE NOWEGO WARIANTU PROBLEMU MARSZRUTYZACJI POJAZDÓW METODĄ CLP

Streszczenie

W artykule przedstawiono nowy wariant problemu marszrutyzacji pojazdów – MDMGVRP, czyli problem marszrutyzacji pojazdów z wieloma magazynami i wieloma asortymentami. Zagadnienie zostało opisane modelem matematycznym a następnie rozwiązane metodą CLP. Opisano dwa programy wykonane przy użyciu narzędzi CLP: Cosytec CHIP i GNU-Prolog. Na koniec przedstawiono przykładowe dane oraz rozwiązanie uzyskane przy pomocy opisanych programów.

Słowa kluczowe: problem marszrutyzacji pojazdów, VRP, MDMGVRP, CLP, CHIP, GNU-Prolog.

INTRODUCTION

In nowadays world of high level of competition there is extremely important to hold a strict regime of cost control. Especially parts of cost that do not bring any added value to a products shall be most reduced. A transportation cost is one of the kind.

Control of logistic processes is one of main matters of concern of operational research. The CLP is a method that help to perfect us dealing with the problem.

1. MDMGVRP VS VRP

You face Vehicle Routing Problem every time when you need to determine routes for certain fleet of vehicles which have limited capacity and are located in one or more depots. The vehicles must distribute goods to a clients. Every client has a demand for a certain number of goods. The distances between client locations and depot locations are given. Usually the aim is to minimize sum of total distance traveled by all cars.

About fifty years have elapsed since Dantzig and Ramser [1] introduced VRP but in the last decade VRP was a matter of interest for operational research. Good overview of the problem was presented by Toth and Vigo [2].

A classification of VRP was introduced by Garn [3]. He also proposed quite wide set of bibliography [4] Sometimes TSP with multiple salesmen is classified as VRP [5], but in fact it is much less complicated and is particular instance of VRP. There is quite a lot of VRP variants, all focused on different problem restrictions.

Every day life comes with various situations, that is why we need to formulate new mathematical models and find a new solution methods. MDMGVRP is for Multiple Depots Multiple Goods Vehicle Routing Problem.

In this case demands are determined for certain set of commodities. The additional constraint is that a stores of goods and a vehicles can be located in a certain number of depots. Every depot has limited stock of goods. This are reasons that make new variant of VRP more complicated.

2. MATHEMATICAL MODEL

Let M be a number of cities, V a number of vehicles and A number of commodities (types of goods). We assume that commodities are packed with the same kind of wrapping, so their number is

determined in same units, which are packages of commodity.

The distances between cities are given with matrix *O*:

$$O = [o_{i,j}] \quad i,j = 1,\ldots,M$$

where *i* means start city, *j* means end city.

There are depots in some cities. In depots a limited number of commodities is stored:

$$S = [s_{a,m}] \quad a = 1,\ldots,A; \quad m = 1,\ldots,M; \quad s_{a,m} \in \mathbf{N}$$

where *a* is means commodity, *m* the city where $s_{a,m}$ packages of commodity *a* is stored.

There are some clients located in the cities. Each client has certain demand:

$$D = [d_{a,m}] \quad a = 1,\ldots,A; \quad m = 1,\ldots,M; \quad d_{a,m} \in \mathbf{N}$$

where *a* is means commodity, *m* the city where $d_{a,m}$ packages of commodity *a* is required.

There can not be a depot and a client located in the same city:

$$\forall m \in \{1,\ldots,M\}: \quad \sum_{a=1}^{A} s_{a,m} = 0 \vee \sum_{a=1}^{A} d_{a,m} = 0$$

Each vehicle has certain capacity l_v that is determined with package of commodity unit:

$$l_v \in \mathbf{N} \quad v = 1,\ldots,V$$

and each vehicle has exploitation cost c_v, which is determined as a cost of travel of one distance unit by the vehicle *v:*

$$c_v \in \mathbf{N} \quad v = 1,\ldots,V$$

Each vehicle starts from specified depot:

$$p_v \quad v = 1,\ldots,V; \quad p_v \in \{1,\ldots,M\}$$

where p_v means the city of start of vehicle *v*.

Let us define:

$$X^v = [x^v_{m_1,m_2}] \quad v = 1,\ldots,V; \quad m_1,m_2 = 1,\ldots,M; \quad x^v_{m_1,m_2} \in \{0,1\}$$

where X^v is the matrix of travel of vehicle *v*, that is determined in following way:

$$x^v_{m_1,m_2} = \begin{cases} 1 & \text{if vehicle } v \text{ travels from city } m_1 \text{ to } m_2 \\ 0 & \text{otherwise} \end{cases}$$

Particular vehicle can visit particular city at most once, so:

$$\sum_{m_1=1}^{M} x^v_{m_1,m_2} \le 1 \qquad \forall v \in \{1,\ldots,V\}, \forall m_2 \in \{1,\ldots,M\}$$

That is also known that particular vehicle leaves particular city at most once:

$$\sum_{m_2=1}^{M} x^v_{m_1,m_2} \le 1 \qquad \forall v \in \{1,\ldots,V\}, \forall m_1 \in \{1,\ldots,M\}$$

If particular vehicle takes part in fulfilling the demands, it would leave its start depot. Otherwise it stays at start depot and all its travel matrix elements are equal to 0:

$$\forall v \in \{1,\ldots,V\}: \quad \sum_{m_2=1}^{M} x^v_{p_s,m_2} = 0 \Rightarrow \sum_{m_1=1}^{M}\sum_{m_2=1}^{M} x^v_{v,m_1,m_2} = 0$$

Let Z^v be a matrix of load, defined in following way:

$$Z = [z^v_{a,m}] \quad v = 1,\ldots,V; \quad a = 1,\ldots,A; \quad m = 1,\ldots,M; \quad z^v_{a,m} \in \mathbf{N}$$

where $z^v_{a,m}$ is the number of packages with commodity *a* loaded to vehicle *v* in city *m*.

The total number of units loaded in a particular city can not be greater than a stock in the city (depot):

$$\forall m \in \{1,\ldots,M\}, \forall a \in \{1,\ldots,A\}: \quad \sum_{v=1}^{V} z^v_{a,m} \le s_{a,m}$$

It also have to be less than a capacity of the vehicle:

$$\forall m \in \{1,\ldots,M\}, \forall a \in \{1,\ldots,A\}, \forall v \in \{1,\ldots,V\}: \quad z^v_{a,m} \le l_s$$

The matrix of unload is defined in the same way:

$$R = [r^v_{a,m}] \quad v = 1,\ldots,V; \quad a = 1,\ldots,A; \quad m = 1,\ldots,M; \quad r^v_{a,m} \in \mathbf{N}$$

where $r^v_{a,m}$ is the number of packages with commodity *a* unloaded from vehicle *v* in city *m*.

All demands shall be fulfilled. It means that number of commodities unloaded in a particular city is equal to demand in the city:

$$\forall m \in \{1,\ldots,M\}, \forall a \in \{1,\ldots,A\}: \quad \sum_{v=1}^{V} r^v_{a,m} = d_{a,m}$$

In MDMGVRP (apart from SDVRP) one client need to be served by a single vehicle:

$$\forall m \in \{1,\ldots,M\}, \exists v \in \{1,\ldots,V\}:$$

$$r^v_{a,m} = d_{a,m} \qquad \forall a \in \{1,\ldots,A\}$$

Each vehicle loads in total the same number of commodities as total number of unloaded commodities during its journey:

$$\forall v \in \{1,\ldots,V\}, \forall a \in \{1,\ldots,A\}:$$

$$\sum_{m=1}^{M} z^v_{a,m} = \sum_{m=1}^{M} r^v_{a,m}$$

A vehicle must visit a city to unload or load commodities. The model must define such constraint, so vehicle must travel in or out the city if number of loaded or unloaded commodities is greater than 0. In this model vehicles do not have to return to their start depots. So there is:

$$\forall v \in \{1,\ldots,V\}, \forall m \in \{1,\ldots,M\}:$$

$$z^v_{a,m} > 0 \vee r^v_{a,m} > 0, \quad a = 1,\ldots,A \Rightarrow$$

$$\sum_{m_2=1}^{M} x^v_{m,m_2} = 1 \vee \sum_{m_1=1}^{M} x^v_{m_1,m} = 1$$

Each vehicle can not be overloaded and underloaded in any visited city. This means that a load of vehicle, which is equal to sum of load state in previous city and change of load (load or unload) in current city, must be less than or equal to vehicle capacity and greater than or equal to zero:

$$\forall v \in \{1,\ldots,V\}, \forall m \in \{1,\ldots,M\}:$$

$$0 \le B(v,m) \le l_v$$

where:

$$B(v,m) = \sum_{a=1}^{A}(z^v_{a,m} - r^v_{a,m}) + B(v,m_0) \qquad x^v_{m_0,m} = 1$$

and in the start depot of vehicle *v*:

$$B(s,p_v) = \sum_{a=1}^{A}(z^v_{a,p_s} - r^v_{a,p_s})$$

You need to remember that vehicle shall visit a city to perform load or unload there. You can describe this relation in following way:

$$\sum_{a=1}^{A}(z^{v_0}_{a,m_0} + r^{v_0}_{a,m_0}) \ne 0 \quad v_0 \in \{1,\ldots,V\}, m_0 \in \{1,\ldots,M\} \Rightarrow$$

$$(\exists m \in \{1,\ldots,M\} : x^{v_0}_{m_0,m} = 1 \vee \exists m \in \{1,\ldots,M\} : x^{v_0}_{m,m_0} = 1)$$

Total cost depends on total distance traveled by all vehicles and their exploitation costs, so:

$$F(X^1,\ldots,X^V) = \sum_{v=1}^{V}\sum_{m_1=1}^{M}\sum_{m_2=1}^{M} x^v_{m_1,m_2} * o_{m_1,m_2} * c_s$$

To solve the problem means to find such X^v, Z^v and R^v, that keep all relations defined above. The goal of optimization is to find the solution with minimum value of objective function F.

3. CLP PROGRAMS

Constraint Logic Programing is still not very popular approach however it is flexible and effective.

Two CLP tools have been used to formulate the programs: Cosytec CHIP and GNU Prolog. Both are Prolog descendant with constraint over finite domains functions. They are similar but there are some differences. The syntaxes are slightly different but more important differences are in functionality. CHIP offers more build-in predicates [6] that help a lot to solve even very complicated problems. GNU Prolog has less predicates but offers a possibility of extending features with function libraries written in C. It also accepts nonlinear constraints [7] that was useful to solve MDMGVRP. Another advantage of GNU-Prolog is that the program is distributed freely under GPL license.

The MDMGVRP programs consist of 4 main sections:

1. Entry data and constraints,
2. Main (computing) part,
3. Additional predicates,
4. Distance matrix definition.

First vehicle parameters are defined:

```
L1=15, C1=3, MS1=4,
```

L means capacity, *C* exploitation cost, *MS* start depot. The digits following letters are indexes. The mean number of vehicle. When there are more than one index of variable in the program, the rule of order is that the first index means vehicle, second is for commodity kind, third is for city.

There are stocks determined forth:

```
S=[S1,S2,S3,S4],
SA1=[SA11,SA12,SA13,SA14],
SA2=[SA21,SA22,SA23,SA24],
[...]
S1#=SA11+SA21+SA31+SA41+SA51+SA61,
S2#=SA12+SA22+SA32+SA42+SA52+SA62,
[...]
SA1=[0,0,0,7],
SA2=[0,0,4,2],
```

S1, S2... mean total stocks of all commodities in the cities. *SA21* means stock of commodities *1* in city *2*.

The demands are defined in the same way:

```
D=[D1,D2,D3,D4],
DA1=[DA11,DA12,DA13,DA14],
DA2=[DA21,DA22,DA23,DA24],
```

In the last section of program a distances between the cities are defined. It is done with predicate *odl*:

```
odl(_,0,0).
odl(1,1,0).
odl(1,2,1).
odl(1,3,3).
[...]
```

Predicate odl has three arguments: first is source city, second is destination city and third the distance between them. The first row is used to calculate objective function value in case when vehicle do not travel from some city.

Next line means that the distance from city *1* to city one is *0* distance units. It is obvious. Third line means that distance from city *1* to city *2* is *1*.

Back to the first row it means that distance from any city to dummy city number *0*, which really means that a vehicle stops, is equal to *0* distance units. When a vehicle stops it do not affect with growth of objective function. It is more understandable after definition of travel matrix.

The definitions in first section of the program are mixed with constraints like:

```
ZA123#=<SA23,
```

This example means that loading of vehicle *1* with commodity *2* in city *3* shall be less or equal appropriate stock in the city.

One of most important constraints is:

```
R12*D2#=R12*R12
```

It holds when total unload of vehicle *1* in city *2* is equal to demand in the city or when is equal to *0*. It is useful to define relation which says that single customer shall be served by single vehicle. However it is possible to use above only in GNU Prolog. In CHIP, because there are no nonlinear constraints, you need to write it in a different way:

```
if (R12#\=D2) then (R12#=0),
```

The travel matrix X^v in program is replaced with vector T^v, which size is equal to number of cities. The index of element means a source city, the value of an element means destination city. If the value is equal to *0*, it means that vehicle stops, it does not leave the city.

For example matrix *X*:

$$X = \begin{bmatrix} 0, & 1, & 0, & 0 \\ 0, & 0, & 0, & 1 \\ 1, & 0, & 0, & 0 \\ 0, & 0, & 1, & 0 \end{bmatrix}$$

is replaced with a vector *T*:

$$T = [2, 4, 1, 3]$$

After definitions and constraints the main section of program is located:

```
statistics(real_time,[Czas_s,_]),
fd_minimize((
fd_labeling(RA11),
fd_labeling(RA12),
fd_labeling(RA21),
fd_labeling(RA22),
fd_labeling(ZA11),
fd_labeling(ZA12),
fd_labeling(ZA21),
fd_labeling(ZA22),

niezaladowane_stoja(Z1,T1),
niezaladowane_stoja(Z2,T2),
jedz_gdzie_trzeba(Z1,R1,T1,MS1),
jedz_gdzie_trzeba(Z2,R2,T2,MS2),

fd_labeling(T1),
fd_labeling(T2),
sprawdz_przeladowanie(MS1,T1,Z1,R1,0,L1),
bilans(MS1,T1,ZA11,RA11,0),
bilans(MS1,T1,ZA12,RA12,0),
sprawdz_przeladowanie(MS2,T2,Z2,R2,0,L2),
bilans(MS2,T2,ZA21,RA21,0),
bilans(MS2,T2,ZA22,RA22,0),
```

```
        dlugosc_trasy(T1,1,DT1),
        dlugosc_trasy(T2,1,DT2),
       CEL is (DT1*C1)+(DT2*C2),

  statistics(real_time,[Czas_l,_]),
           write('Cel'),nl,
            write(CEL),nl,
         write('EUREKA!'),nl,
        write('Zaladunki'),nl,
     write('Z1='),write(Z1),nl,
   write('ZA11='),write(ZA11),nl,
   write('ZA12='),write(ZA12),nl,
     write('Z2='),write(Z2),nl,
   write('ZA21='),write(ZA21),nl,
   write('ZA22='),write(ZA22),nl,
       write('Rozladunki'),nl,
     write('R1='),write(R1),nl,
   write('RA11='),write(RA11),nl,
   write('RA12='),write(RA12),nl,
     write('R2='),write(R2),nl,
   write('RA21='),write(RA21),nl,
   write('RA22='),write(RA22),nl,
         write('Trasy'),nl,
     write('T1='),write(T1),nl,
     write('T2='),write(T2),nl,
           write(''),nl,
       Czas is Czas_l-Czas_s,
      write('Czas liczenia='),
    write(Czas),write(' ms'),nl,
           write(''),nl
                ),
               CEL
                ),
```

In GNP Prolog the main role plays predicate *fd_minimize*. It runs a code given as first parameter until the second parameter (in this case variable CEL) achieves minimum value.

The first part of code inside *fd_mnimize* is labeling of loads and unloads. Then there are some additional constrains in predicates *niezaladowane_stoja* and *jedz_gdzie_trzeba.* They provide that every vehicle that do not fulfill any demand stays in start depot and that every city where loads or unloads are performed is visited by the appropriate vehicle.

Then there is labeling of vectors Tv and predicates sprawdz_przeladowanie and bilans. The predicates provide that all vehicle routes are continuous. Vector that contain two routes like *T=[2,1,0,5,4]* are forbidden. The predicates also provide balance checking for routs. It means that any vehicle can not be over or underload all route long.

4. DEMONSTRATIONAL PROBLEM

Consider MDMGVRP with given 2 vehicles, 2 commodity kinds and 6 cities. There are two depots and four customers.

Below stock values are defined:

```
SA1=[0,0,0,7,2,0],
SA2=[0,0,0,4,0,0],
```

and demand values:

```
DA1=[2,4,1,0,0,0],
DA2=[1,2,0,0,0,0],
```

Vehicle parameters are following:

```
L1=10, C1=3, MS1=6,
L2=6, C2=1, MS2=6,
```

The distance matrix:

$$O = \begin{bmatrix} 0, & 1, & 3, & 2, & 4, & 5 \\ 1, & 0, & 1, & 2, & 3, & 4 \\ 3, & 1, & 0, & 1, & 3, & 3 \\ 2, & 2, & 1, & 0, & 1, & 2 \\ 4, & 3, & 3, & 1, & 0, & 1 \\ 5, & 4, & 3, & 2, & 1, & 0 \end{bmatrix}$$

GNU Prolog that is running on machine with Intel processor (Celeron 1.4GHz, 256MB RAM, OS MS Windows XP Home) solves the problem above in time of 32397 milliseconds and shows the solution:

```
                Cel
                15
EUREKA! Rozwiazanie ostateczne:
              Zaladunki
            Z1=[0,0,0,8,2,0]
           ZA11=[0,0,0,5,2,0]
           ZA12=[0,0,0,3,0,0]
            Z2=[0,0,0,0,0,0]
           ZA21=[0,0,0,0,0,0]
           ZA22=[0,0,0,0,0,0]
              Rozladunki
            R1=[3,6,1,0,0,0]
           RA11=[2,4,1,0,0,0]
           RA12=[1,2,0,0,0,0]
            R2=[0,0,0,0,0,0]
           RA21=[0,0,0,0,0,0]
           RA22=[0,0,0,0,0,0]
                Trasy
            T1=[0,1,2,3,4,5]
            T2=[0,0,0,0,0,0]
         Czas liczenia=32397 ms
```

It also signalize that there are more optimal solutions (with the same objective function value). It is possible to view them all.

CHIP needs time of 38505 milliseconds for completing the task and shows only the first solution found.

5. SUMMARY

CLP is useful method to solve MDMGVRP a new variant of VRP and quite complicated one.

MDMGVRP is still an issue of research as there is a need to determine maximal size of the problem that can be solved with CHIP and GNU-Prolog.

The CLP solution should be compared with some other methods and maybe performance should improved by using heuristic methods or other means.

REFERENCES

[1] Dantzig G. B., Ramser R. H. *The Truck Dispatching Problem*. Management Science, 1959, Vol. 6, pp. 80–91.

[2] Toth P., Vigo D. *The Vehicle Routing Problem.* SIAM Monographs on Discrete Mathematics and Applications, 2002, pp. 1–25.

[3] Garn W.: *http://osiris.tuwien.ac.at/~wgarn/ VehicleRouting/vehicle_routing.html*. Internet pucation, Wien, 2002.

[4] Garn W.: *http://osiris.tuwien.ac.at/~wgarn/ VehicleRouting/neo/biblio.html*. Internet pucation, Wien, 2002.

[5] Jadczak R., Trzaskalik T.: *Algorytmy genetyczne, ewolucyjne i metaheurystyki. Informatyka w badaniach operacyjnych*, Prace naukowe AE, Katowice, 2005, pp. 79-111.
[6] COSYTEC SA.: *CHIP System documentation.* Orsay Cedex, 1997.
[7] Diaz D.: *GNU Prolog documentation*, Paris, 2002.

Rafał SZKLARCZYK is a graduate of Academy of Computer Science and Management in Bielsko-Biała (computer science engineer) and Academy of Mining and Metallurgy in Cracow (master of management). Now he is running a software engineering company with success for over five years. He is also a PhD student at Silesian University of Technology with special interest on operational research and CPL methods.