Warehouse Location
(Project presentation of 5th August 2022)

Clara CLÄVER and Whiz KIDD
Team t

Uppsala University
Sweden

Course 1DL442:
Combinatorial Optimisation and Constraint Programming,
whose part 1 is Course 1DL451:
Modelling for Combinatorial Optimisation

UPPSALA UNIVERSITET

Outline

UPPSALA UNIVERSITET

Outline

1. **Problem**
2. Example
3. Approach
4. Experiments
5. Conclusion

UPPSALA UNIVERSITET

Warehouse Location: Setting and Data

A company considers opening warehouses at some candidate locations in order to supply its existing shops:

UPPSALA UNIVERSITET

Warehouse Location: Setting and Data

A company considers opening warehouses at some candidate locations in order to supply its existing shops:

- Each candidate warehouse here has (for simplicity) the same maintenance cost.

UPPSALA UNIVERSITET

Warehouse Location: Setting and Data

A company considers opening warehouses at some candidate locations in order to supply its existing shops:

- Each candidate warehouse here has (for simplicity) the same maintenance cost.
- Each candidate warehouse has a supply capacity, which is the maximum number of shops it can supply.

UPPSALA UNIVERSITET

Warehouse Location: Setting and Data

A company considers opening warehouses at some candidate locations in order to supply its existing shops:

- Each candidate warehouse here has (for simplicity) the same maintenance cost.
- Each candidate warehouse has a supply capacity, which is the maximum number of shops it can supply.
- The supply cost to a shop depends on the warehouse.

[In general: a picture is worth a thousand words!]

UPPSALA UNIVERSITET

Warehouse Location: Problem

Determine which candidate warehouses actually to open, and which of them supplies which shops, so that:

UPPSALA UNIVERSITET

Warehouse Location: Problem

Determine which candidate warehouses actually to open, and which of them supplies which shops, so that:

1Supplier: Each shop is supplied by exactly one actually opened warehouse.

UPPSALA UNIVERSITET

Warehouse Location: Problem

Determine which candidate warehouses actually to open, and which of them supplies which shops, so that:

1Supplier: Each shop is supplied by exactly one actually opened warehouse.

Capacity: Each actually opened warehouse supplies a number of shops at most equal to its capacity.

UPPSALA UNIVERSITET

Warehouse Location: Problem

Determine which candidate warehouses actually to open, and which of them supplies which shops, so that:

1Supplier: Each shop is supplied by exactly one actually opened warehouse.

Capacity: Each actually opened warehouse supplies a number of shops at most equal to its capacity.

MinCost: The sum of the actually incurred maintenance costs and supply costs is minimal.

[In general: a picture is worth a thousand words!]

UPPSALA UNIVERSITET

Outline

UPPSALA UNIVERSITET

Simple data:

```
Warehouses = {Bern, Cork, Lyon, Oslo, Roma};
Shops = {Shop1, Shop2, Shop3, ..., Shop10};
% Each warehouse has the same maintenance cost:
maintCost = 30;
% Capacity[w] = supply capacity of warehouse w:
Capacity = [1,4,2,1,3];
% SupplyCost[s,w] = supply cost to s from w:
SupplyCost =
  [| 20, 24, 11, 25, 30 |
     28, 27, 82, 83, 74 |
     ...
     47, 65, 55, 71, 95 |];
```

Simple data:

```
Warehouses = {Bern, Cork, Lyon, Oslo, Roma};
Shops = {Shop1, Shop2, Shop3, ..., Shop10};
% Each warehouse has the same maintenance cost:
maintCost = 30;
% Capacity[w] = supply capacity of warehouse w:
Capacity = [1,4,2,1,3];
% SupplyCost[s,w] = supply cost to s from w:
SupplyCost =
  [| 20, 24, 11, 25, 30 |
     28, 27, 82, 83, 74 |
     ...
     47, 65, 55, 71, 95 |];
```

The minimal cost is 383, uniquely for: Bern supplies shop 4; Cork supplies shops 2, 6, 7, 9; Lyon supplies shops 8, 10; Oslo supplies no shops; and Roma supplies shops 1, 3, 5.

UPPSALA UNIVERSITET

Simple data:

```
2  Warehouses = {Bern, Cork, Lyon, Oslo, Roma};
3  Shops = {Shop1, Shop2, Shop3, ..., Shop10};
4  % Each warehouse has the same maintenance cost:
5  maintCost = 30;
6  % Capacity[w] = supply capacity of warehouse w:
7  Capacity = [1,4,2,1,3];
8  % SupplyCost[s,w] = supply cost to s from w:
9  SupplyCost =
10   [| 20, 24, 11, 25, 30 |
11      28, 27, 82, 83, 74 |
12      ...
13      47, 65, 55, 71, 95 |];
```

The minimal cost is 383, uniquely for: Bern supplies shop 4; Cork supplies shops 2, 6, 7, 9; Lyon supplies shops 8, 10; Oslo supplies no shops; and Roma supplies shops 1, 3, 5.

Our experiments at slide 20 use hard third-party real data from https://www.csplib.org/Problems/prob034/data.

UPPSALA UNIVERSITET

Outline

Parameters

```
2 enum Warehouses;
3 enum Shops;
4 % Each warehouse has the same maintenance cost:
5 int: maintCost;
6 % Capacity[w] = supply capacity of warehouse w:
7 array[Warehouses] of int: Capacity;
8 % SupplyCost[s,w] = supply cost to s from w:
9 array[Shops,Warehouses] of int: SupplyCost;
```

[In general: present *only* your *best* approach so far!]

UPPSALA UNIVERSITET

Decision Variables and Channelling

Automatic enforcement of the 1Supplier constraint (each shop is supplied by exactly 1actually opened warehouse):

Decision Variables and Channelling

Automatic enforcement of the 1Supplier constraint (each shop is supplied by exactly 1actually opened warehouse):

```
10 % Supplier[s] = the supplier warehouse for s:
11 array[Shops] of var Warehouses: Supplier;
```

Decision Variables and Channelling

Automatic enforcement of the 1Supplier constraint (each shop is supplied by exactly 1actually opened warehouse):

```
10 % Supplier[s] = the supplier warehouse for s:
11 array[Shops] of var Warehouses: Supplier;
```

For expressing the search strategy (at slide 16), we *need* non-mutually redundant variables, and we channel 1-way:

```
13 % Cost[s] = actually incurred supply cost for s:
14 array[Shops] of var 0..max(SupplyCost): Cost;
15 constraint forall(s in Shops) % 1way channelling
16   (Cost[s] = SupplyCost[s,Supplier[s]]);
```

[In general: (experimentally) justify the use or absence of (mutually or non-mutually) redundant variables and their (one-way or two-way) channelling constraints!]

UPPSALA UNIVERSITET

Objective

We model the objective function (the sum of the actually incurred maintenance costs and supply costs is to be minimised) using the redundant variables for brevity:

Objective

We model the objective function (the sum of the actually incurred maintenance costs and supply costs is to be minimised) using the redundant variables for brevity:

```
25   maintCost * nvalue(Supplier) %   total maint.
26   + sum(Cost);                 % + total supply
```

[In general: crosscheck your model against the advice of Topics 2 and 3 at http://user.it.uu.se/~pierref/courses/COCP/demoReport/checklist.pdf.]

UPPSALA UNIVERSITET

Capacity Constraint and Inference

Capacity constraint (each actually opened warehouse supplies a number of shops at most equal to its capacity):

Capacity Constraint and Inference

Capacity constraint (each actually opened warehouse supplies a number of shops at most equal to its capacity):

```
18 constraint global_cardinality_low_up_closed
19    (Supplier, Warehouses,
20     [0 | w in Warehouses], Capacity)
21    :: domain_propagation;
```

Capacity Constraint and Inference

Capacity constraint (each actually opened warehouse supplies a number of shops at most equal to its capacity):

```
18 constraint global_cardinality_low_up_closed
19   (Supplier, Warehouses,
20    [0 | w in Warehouses], Capacity)
21   :: domain_propagation;
```

We suggest domain consistency to CP and LCG backends, as our tests show their defaults are slower on hard data.

[In general: (experimentally) justify inference annotations!]

UPPSALA UNIVERSITET

Implied Constraints

We did not yet derive any useful implied constraints.

[In general: (experimentally) justify the use or absence of implied constraints and their inference annotations!]

UPPSALA UNIVERSITET

Symmetry-Breaking Constraints

We did not detect any symmetries in the problem or model.

Symmetry-Breaking Constraints

We did not detect any symmetries in the problem or model.

There can be (possibly dynamic) instance symmetries — namely warehouses that (dynamically) have the same capacity (whereas shops with the same supply costs are extremely unlikely) — but the supply costs of same-capacity warehouses most likely differ in real life, so we decided not to detect and exploit instance symmetries.

[In general: (experimentally) justify the use or absence of symmetry-breaking constraints and their inference annotations!]

UPPSALA UNIVERSITET

Output

In order to display intermediate objective values on-the-fly, we need another non-mutually redundant variable, and we channel it 1-way to take the objective value (see slide 11):

UPPSALA UNIVERSITET

Output

In order to display intermediate objective values on-the-fly, we need another non-mutually redundant variable, and we channel it 1-way to take the objective value (see slide 11):

```
23 var 0..(maintCost*card(Warehouses) +
     sum(s in Shops)(max(SupplyCost[s,..]))): cost;
24 constraint cost = % one-way channelling:
25   maintCost * nvalue(Supplier) %   total maint.
26   + sum(Cost);                 % + total supply
```

Output

In order to display intermediate objective values on-the-fly, we need another non-mutually redundant variable, and we channel it 1-way to take the objective value (see slide 11):

```
23 var 0..(maintCost*card(Warehouses) +
     sum(s in Shops)(max(SupplyCost[s,..]))): cost;
24 constraint cost = % one-way channelling:
25   maintCost * nvalue(Supplier) %   total maint.
26   + sum(Cost);                 % + total supply
```

so that the objective becomes

```
solve minimize cost;
```

Output

In order to display intermediate objective values on-the-fly, we need another non-mutually redundant variable, and we channel it 1-way to take the objective value (see slide 11):

```
23 var 0..(maintCost*card(Warehouses) +
     sum(s in Shops)(max(SupplyCost[s,..]))): cost;
24 constraint cost = % one-way channelling:
25   maintCost * nvalue(Supplier) %   total maint.
26   + sum(Cost);                 % + total supply
```

so that the objective becomes

```
    solve minimize cost;
```

and the output statement becomes

```
34 output ["Cost = \(cost) for Supplier = \(Supplier)"];
```

UPPSALA UNIVERSITET

Search

We suggest the maximal-regret search strategy to CP, LCG, and MIP backends. It is expressed as follows:

UPPSALA UNIVERSITET

Search

We suggest the maximal-regret search strategy to CP, LCG, and MIP backends. It is expressed as follows:

```
28 solve
29  :: seq_search([
30       int_search(Cost,max_regret,indomain_min),
31       int_search(Supplier,input_order,indomain_min)
32      ])
33   minimize cost;
```

UPPSALA UNIVERSITET

Search

We suggest the maximal-regret search strategy to CP, LCG, and MIP backends. It is expressed as follows:

```
28 solve
29  :: seq_search([
30       int_search(Cost,max_regret,indomain_min),
31       int_search(Supplier,input_order,indomain_min)
32     ])
33   minimize cost;
```

We experimentally established the merit of this strategy (over the default and other strategies) for this model in the slides of Topic 8.

[In general: (experimentally) justify search annotations!]

UPPSALA UNIVERSITET

Efficiency

The model features no violations of any pieces of advice in the checklists of Topics 2 and 3.

[In general: For each violation of a piece of advice of Topics 2 and 3 in `http://user.it.uu.se/~pierref/courses/COCP/demoReport/checklist.pdf`: how do you argue that it does not matter? For example, for a reification or for a `where` clause involving variables, does a profiled compilation reveal numbers of generated variables and constraints that you argue to be acceptable; or is the solving time comparable to the one of a violation-free reformulation that you give?]

UPPSALA UNIVERSITET

Correctness

All the objective values reported in the experiments that were proven minimal before timing out do not contradict those found by the approach of Team 11.

[In general: How do you argue for the correctness of your approach? For example, did you use a checker based on another model, say a model of another team or a translation of a MIP model given somewhere? Did you use the same instances as another team? If it is a satisfaction problem: did you compare the numbers of solutions with numbers reported somewhere or by another team? If it is an optimisation problem: did you compare objective values proven optimal before timing out (respectively objective values known when timing out) with optimal objective values (respectively best-known objective values) reported somewhere or by another team?]

UPPSALA UNIVERSITET

Outline

UPPSALA UNIVERSITET

The hard real-life instance `cap44` at CSPlib.org supplies 50 shops from 16 candidate warehouses, all of capacity 4. We did not find a published minimal total cost.

The hard real-life instance `cap44` at CSPlib.org supplies 50 shops from 16 candidate warehouses, all of capacity 4. We did not find a published minimal total cost.

Results within 600 seconds on an iMac (late 2013, 3.2 GHz Intel Core i5, RAM 8 GB 1600 MHz DDR3):

Backend	Gecode		CP-SAT		Gurobi		Yuck		PicatSAT	
data	obj	time	obj	time	obj	time	obj	time	obj	time
cap44	1193	t/o	**1190**	t/o	1193	t/o	1198	t/o	1218	t/o
cap63	...	...	...	...	...	...	...	...	...	...
cap71	...	...	...	...	...	...	...	...	...	...
cap81	...	...	...	...	...	...	...	...	...	...
cap101	...	...	...	...	...	...	...	...	...	...
cap131	...	...	...	...	...	...	...	...	...	...

(Clara and Whiz will run the remaining experiments later)

[In general: tweak & evaluate the approach on backends of all the considered technologies, until at least two backends of distinct technologies are at least somewhat competitive with the state of the art, and no backend produces errors.]

UPPSALA UNIVERSITET

Outline

UPPSALA UNIVERSITET

Conclusion

Insights:

- . . .

UPPSALA UNIVERSITET

Conclusion

Insights:

- . . .
- . . .

UPPSALA UNIVERSITET

Conclusion

Insights:

- . . .
- . . .

Future work for the final project report:

- . . .

UPPSALA UNIVERSITET

Conclusion

Insights:

- . . .
- . . .

Future work for the final project report:

- . . .
- . . .

UPPSALA UNIVERSITET

Conclusion

Insights:

- . . .
- . . .

Future work for the final project report:

- . . .
- . . .