UNIVERSITY OF LISBON
INSTITUTO SUPERIOR TÉCNICO
MSC. COMPUTER SCIENCE AND ENGINEERING

TÉCNICO LISBOA

Algorithms for Computational Logic

Project 1 - Report

Harvesting Scheduling

90775 - Ricardo Fernandes

LISBON, OCTOBER 2022

1 Encoding

The encoding of the problem to SAT variables was thought with the help of the following tables:

E	U_1	U_2	U_i	U_n
T_0	1	2	i	n
T_1	$n+1$	$n+2$	$n+i$	$2 \cdot n$
T_2	$2 \cdot n+1$	$2 \cdot n+2$	$2 \cdot n+i$	$3 \cdot n$
T_j	$j \cdot n+1$	$j \cdot n+2$	$\mathbf{j \cdot n+i}$	$(j+1) \cdot n$
T_k	$k \cdot n+1$	$k \cdot n+2$	$k \cdot n+i$	$(k+1) \cdot n$
T_{k+1}	$(k+1) \cdot n+1$	$(k+1) \cdot n+2$	$(k+1) \cdot n+i$	$(k+2) \cdot n$

T_0 is when a unit is neither harvested or a natural reserve, T_{k+1} is when a unit is a natural reserve, this way it simplifies the constraints because there is no need to specify that when a unit is a natural reserve it cannot be harvested in any period. This table can be translated with the following formula:

This encoding is enough for the whole problem except the contiguity part. Auxiliary variables are needed to guarantee the singular contiguity of the natural reserve units. This is because depth information will be required when creating the problem constraints (more information in sections ahead). Contiguity does not require harvesting period information, only unit id information, therefore we obtain the SAT encoding with the table below:

E	U_1	U_2	U_i	U_n
D_1	$(k+2) \cdot n+1$	$(k+2) \cdot n+2$	$(k+2) \cdot n+i$	$(k+3) \cdot n$
D_2	$(k+3) \cdot n+1$	$(k+3) \cdot n+2$	$(k+3) \cdot n+i$	$(k+4) \cdot n$
D_d	$(k+1+d) \cdot n+1$	$(k+1+d) \cdot n+2$	$\mathbf{(k+1+d) \cdot n+i}$	$(k+2+d) \cdot n$
D_m	$(k+1+m) \cdot n+1$	$(k+1+m) \cdot n+2$	$(k+1+m) \cdot n+i$	$(k+2+m) \cdot n$

Because SAT variables can't be repeated, the first cell of the table starts at the value of the previous table last cell plus one: $(k+2) \cdot n+1$.

The maximum depth a natural reserve can have, m, is the length of the maximum areas needed to be just greater than the natural reserve minimum area A_{min}. More about this, check optimization 2.

The encoding of the table can then be defined as the following formula:

$$E(i,d) = (k+1+d) \cdot n + i \qquad i \in U, \qquad d \in D = \{1, ..., m\}$$

Merging the two tables and their formulas, we get the following encoding function for the problem:

$$E(i,r) = r \cdot n + 1, \qquad i \in U = \{1, ..., n\}, \qquad r \in \{0, ..., k+2+m\}$$

$$r = \begin{cases} 0 & \text{not harvested} & \text{(table 1)} \\ j & r \in \{0, ..., k+1\}, \quad j \in T & \text{(table 1)} \\ k+1 & \text{natural reserve} & \text{(table 1)} \\ k+1+d & r \in \{k+2, ..., k+2+m\}, \quad d \in D & \text{(table 2)} \end{cases}$$

2 Constraints

Consider the following definitions for the next formulas:

$U = \{1, ..., n\}, \quad T = \{1, ..., k\}, \quad D = \{1, ..., m\},$
$D_R = \{D_1, D_2, ..., D_m\} = \{k+2, k+3, ..., k+1+m\}$
$A = [A_1, A_2, ..., A_n]$: Areas for each unit
$P(i, j)$: returns the profit of the unit i at period j

$$E(i,j) \rightarrow \neg E(i,j') \qquad \forall_{i \in U} \forall_{j \in T \cup \{0\}} \forall_{j' \in [j+1, k+1]} \tag{1}$$

$$E(i,j) \rightarrow \neg E(i_n,j) \qquad i_n > i, \ \ \forall_{i \in U} \forall_{i_n \in i_{neighbours}} \forall_{j \in T} \tag{2}$$

$$\sum_{i \in U} A_i \cdot E(i, k+1) \geq A_{min} \tag{3}$$

$$\sum_{i \in U} E(i, D_1) \leq 1 \tag{4.1}$$

$$\sum_{d \in D_R} E(i,d) \leq 1 \qquad \forall i \in U \tag{4.2}$$

$$E(i,d) \rightarrow \Big(\sum_{i_n \in i_{neighbours}} E(i_n, d-1) = 1 \Big) \qquad \forall_{i \in U} \forall_{d \in D_R \setminus \{k+2\}} \tag{4.3}$$

$$E(i, k+1) \leftrightarrow \sum_{d \in D_R} E(i,d) \geq 1 \qquad \forall_{i \in U} \tag{4.4}$$

$$\max \sum_{i \in U} \sum_{j \in T} E(i,j) \cdot P(i,j) \tag{5}$$

1. **A unit of land can only be harvested once.** If unit i is harvested at the period j, i cannot be harvested at another period j'. Check optimization 3.

2. **Neighbour units cannot be harvested in the same time period.** If unit i is harvested at the period j, then every neighbour of i, i_n cannot be harvested at that same period. Check optimization 4.

3. **The natural reserve area must be $\geq A_{min} \geq 0$.** The sum of areas of each natural reserve unit (period $k+1$) must be greater than the established minimum natural reserve area.

4. **The natural reserve area is contiguous.**

   As mentioned before, singular contiguity can only be defined with the notion of depth of the contiguity, this is accomplished with the help of auxiliary variables.

   We can think of it as a uni-directional tree, where the root is the start of the contiguity. With the following constraints, it's possible to solve the contiguity problem.

   4.1 **There can only be one root.** The sum of units in tree at depth 1 must be at most 1.

   4.2 **A unit must either not belong to the tree or only be present in one level in the tree.** For unit i, the sum of presences in the tree at different layers must be at most 1.

   4.3 **A unit in the tree has to have one, and can only have one, predecessor.** If unit i is present in the tree at depth d, not being the root level 1, then, the sum of presences of i neighbours, i_n, in the tree at depth $d-1$ must be exactly 1.

4.4 **A natural reserve unit must be in the tree and a unit that is in the tree is a natural reserve unit.** If unit i is a natural reserve (period $k+1$) then, then sum of the presences of i in the different depths of the tree must be at least 1. And also the reverse, meaning, if i at least is present in one depth of tree then, it must be a natural reserve.

5. **Maximize the sum of profits when the units were harvested.**

3 Implementation

The implementation was done with the programming language python, version 3.10, with the libraries `python-sat>=0.1.7.dev19` and `pypblib>=0.0.4`.

The **Pseudo-Boolean encoding** was used to solve the (3) constraint. **Cardinality constraints** with **sequential counter** was used solve all the ***at most*** and ***equal*** constraints, (4.1), (4.2) and (4.3). The ***at least*** was simply solved by adding a clause with all the variables as a disjunction. Partially weighted **MaxSAT** was used to maximize the profit (5) adding each pair unit, period as soft clauses with its respective profit as the weight.

4 Optimizations

1. The encodings were being constantly optimized to lower redundancy and empty SAT variables.

2. The maximum depth a natural reserve can have, m, was initially defined as n, the total number of units, and after optimized to length of the maximum areas needed to be just greater than the natural reserve minimum area A_{min}. Pseudo-code:

```
max_depth = 0
nr_min_area_sum = 0
for i_area in sorted(areas):
    nr_min_area_sum += i_area
    max_depth += 1
    if nr_min_area_sum >= amin:
        break
```

3. In constraint (1), instead of j' starting at 1 *like* j, it starts at $j+1$, because this way, less iterations are ran and no redundant constraints are added. For example:

$$E(U_1,T_1) \rightarrow \neg E(U_1,T_3) \Leftrightarrow \neg E(U_1,T_1) \vee \neg E(U_1,T_3)$$
$$E(U_1,T_3) \rightarrow \neg E(U_1,T_1) \Leftrightarrow \neg E(U_1,T_3) \vee \neg E(U_1,T_1)$$

   As we can see above the two constraints are the same, this happens if j' value starts below j value.

4. Similar to optimization 1, in constraint (2) instead of going through every neighbour of i, to avoid redundant constraints, it's only needed to go through neighbours whose id is greater than i. For example, let's say U_3 is neighbour of U_2, in the iteration where $i = 2$ it will

add the clause $E(2,j) \rightarrow \neg E(3,j) \Leftrightarrow \neg E(2,j) \vee \neg E(3,j), \forall_{j \in T}$; and in the iteration $i = 3$, if there the optimization didn't exist, it would add also the clause $E(3,j) \rightarrow \neg E(2,j) \Leftrightarrow \neg E(3,j) \vee \neg E(2,j), \forall_{j \in T}$. As we can see, these two clauses are the same.

5. If the natural reserve minimum area is zero, the (3) and (4) constraints are not added.

6. The performance was dramatically increased when, in the solver, adapt intrinsic was turned on with a flag in 'RC2' class.

5 Run

To run the project, install the dependencies: `pip3 install -r requirements.txt`, and then run the main python file: `python3 main.py` or just run `./proj1`

References

[1] Tiago Almeida and Vasco Manquinho. Constraint-based electoral districting using a new compactness measure: An application to portugal. *Computers & Operations Research*, 146:105892, 2022.