1 Explanation Generation

Algorithm 1: CSP-Explain($\mathcal{T}$, f [, $\mathcal{I}_0$])

input : $\mathcal{T}$ *set of constraints*
input : f *a consistent objective function*
optional input: $\mathcal{I}_0$ *a partial interpretation*
output : *Explanation sequence*

begin
  $\mathcal{I}_{end} \leftarrow$ `propagate(`$\mathcal{I}_0$, $\mathcal{T}$`)` // Goal state
  $\mathcal{I} \leftarrow \mathcal{I}_0$ // Initial partial interpretation
  $Seq \leftarrow \emptyset$ // explanation sequence
  while $\mathcal{I} \neq \mathcal{I}_{end}$ **do**
    for $i \in \mathcal{I}_{end} \setminus \mathcal{I}$ **do**
      $X_i \leftarrow$ `OMUS(`$\{\neg i\} \wedge \mathcal{I} \wedge \mathcal{S}$`)`
      $E_i \leftarrow \mathcal{I} \cap X_i$ // Facts used
      $S_i \leftarrow \mathcal{T} \cap X_i$ // Constraint used
      $\mathcal{N}_i \leftarrow$ `propagate(`$E_i \wedge \mathcal{S}_i$`)` // Newly derived facts
    end
    $(E_{best}, S_{best}, N_{best}) \leftarrow (E_i, S_i, N_i)$ with lowest $f(E_i, S_i, N_i)$
    append $(E_{best}, S_{best}, N_{best})$ to Seq
    $\mathcal{I} \leftarrow \mathcal{I} \cup \{N_{best}\}$
  end
end

Algorithm 2: CSP-Explain-Incremental($\mathcal{T}$, f [, $\mathcal{I}_0$])

$\mathcal{I}_{end} \leftarrow$ `propagate(`$\mathcal{I}_0$, $\mathcal{T}$`)`
$\mathcal{I} \leftarrow \mathcal{I}_0$
$, \mathcal{W} \leftarrow \emptyset$ `// Collection of MSSes generated during OMUS`
$Seq \leftarrow \emptyset$
while $\mathcal{I} \neq \mathcal{I}_{end}$ **do**
    for $i \in \mathcal{I}_{end} \setminus \mathcal{I}$ **do**
        X_i, $\mathcal{MSS} \leftarrow$ `OMUS-DELAYED(`$\{\neg i\} \wedge \mathcal{I} \wedge \mathcal{S}$, $\mathcal{M}$`)`
        $E_i \leftarrow \mathcal{I} \cap X_i$
        $S_i \leftarrow \mathcal{T} \cap X_i$
        $\mathcal{N}_i \leftarrow$ `propagate(`$E_i \wedge \mathcal{S}_i$`)`
        $\mathcal{M} \leftarrow \mathcal{M} \cup \mathcal{MSS}$
    end
    $(E_{best}, S_{best}, N_{best}) \leftarrow (E_i, S_i, N_i)$ with lowest $f(E_i, S_i, N_i)$
    append $(E_{best}, S_{best}, N_{best})$ to Seq
    $\mathcal{I} \leftarrow \mathcal{I} \cup \{N_{best}\}$
end

2 OMUS Algorithm

Note that if we assign a unit weight to every element in the subset, we reduce the problem of finding an OMUS to finding a SMUS.

Definition 1. *Let Γ be a collection of sets and HS(Γ) the set of all hitting sets on Γ and let f be an valid objective function. Then a hitting set $h \in \Gamma$ is said to be an **optimal** hitting set if $\forall\ h' \in HS(\Gamma)$ we have that $f(h) \leq f(h')$.*

Property 1. *The **optimal** hitting set of a collection of sets Γ is denoted by $OHS(\Gamma)$.*

The algorithm is based on the following observation:

Proposition 1. *A set $\mathcal{U} \subseteq \mathcal{F}$ is an OMUS of $\mathcal{F}$ if and only if $\mathcal{U}$ is an optimal hitting set of MCSes($\mathcal{F}$)*

Lemma 1. *Let $\mathcal{K} \subseteq$ MCSes($\mathcal{F}$). Then a subset $\mathcal{U}$ of $\mathcal{F}$ is an OMUS if $\mathcal{U}$ is a optimal hitting set on $\mathcal{K}$ and $\mathcal{U}$ is unsatisfiable*

Algorithm 3: `OMUS-Delayed`($\mathcal{F}, \mathcal{M}, f_{cost}$)

```
// F = unsatisfiable CNF formula; M = Collection of MSSes; f_cost = cost function
K ← ∅
// grow mss from input mss
foreach MSS ∈ M do
    MSS' ← Grow(F ∩ MSS)
    K ← K ∪ {F \ MSS'}
end
mode ← mode_greedy
while true do
    while true do
        switch nonOptLevel do
            case mode_incr
                hs ← FindIncrementalHittingSet(K, C, hs)
            case mode_greedy
                hs ← FindGreedyHittingSet(K)
        end
        (sat?, κ) ← SatSolver(hs)
        if not sat? then
            switch nonOptLevel do
                case mode_incr
                    mode ← mode_greedy
                case mode_greedy
                    mode ← mode_opt
                    break
            end
        else
            MSS ← Grow(hs)
            M ← M ∪ {MSS}
            K ← K ∪ {F \ MSS}
            mode ← mode_incr
    end
    hs ← OptimalHittingSet(K, f_cost)
    (sat?, κ) ← SatSolver(hs)
    if not sat? then
        return hs, M
    end
    MSS ← Grow(hs)
    M ← M ∪ {MSS}
    K ← K ∪ {F \ MSS}
    mode ← mode_incr
end
```

Algorithm 4: `OMUS-Delayed`$(\mathcal{F}, \mathcal{M}, f_{cost})$

`// F = unsatisfiable CNF formula; M = Collection of MSSes;` f_{cost} `= cost function`
$\mathcal{K} \leftarrow \emptyset$
`// grow mss from input mss`
foreach $\mathcal{MSS} \in \mathcal{M}$ **do**
    $\mathcal{MSS}' \leftarrow$ `Grow`$(\mathcal{F} \cap \mathcal{MSS})$
    $\mathcal{K} \leftarrow \mathcal{K} \cup \{\mathcal{F} \setminus \mathcal{MSS}'\}$
end
mode $\leftarrow$ mode_greedy
while *true* **do**
    switch *nonOptLevel* **do**
        case *mode_incr*
            $\mathcal{MCS} \leftarrow \{\mathcal{F} \setminus MSS\}$
            $hs \leftarrow$ `FindIncrementalHittingSet`$(\mathcal{K}, \mathcal{MCS}, hs)$
        case *mode_greedy*
            $hs \leftarrow$ `FindGreedyHittingSet`$(\mathcal{K})$
        case *mode_opt*
            $hs \leftarrow$ `OptimalHittingSet`$(\mathcal{K}, \mathcal{W})$
    end
    (sat?, κ) $\leftarrow$ `SatSolver`(hs)
    if *not sat?* **then**
        switch *nonOptLevel* **do**
            case *mode_incr*
                mode $\leftarrow$ mode_greedy
            case *mode_greedy*
                mode $\leftarrow$ mode_opt
                break
            case *mode_opt*
                return *hs*, $\mathcal{M}$
        end
    else
        $MSS \leftarrow$ `Grow`(hs)
        $\mathcal{M} \leftarrow \mathcal{M} \cup \{MSS\}$
        $\mathcal{K} \leftarrow \mathcal{K} \cup \{\mathcal{F} \setminus MSS\}$
        mode $\leftarrow$ mode_incr
end

3 MIP hitting set problem specification

For the set of clauses $\mathcal{C} = \{c_1, ...c_{|C|}\}$ with weights $\mathcal{W} = \{w_1, ...w_{|C|}\}$ in the collection of sets $\mathcal{H}$. For Example:

$$\begin{aligned}
\mathcal{C} &= \{c_1, ...c_6\} \\
\mathcal{W} &= \{w_1 = 20, w_2 = 20, w_3 = 10, w_4 = 10, w_5 = 10, w_6 = 20\} \\
\mathcal{H} &= \{c_3\},\ \{c_2, c_4\},\ \{c_1, c_4\},\ \{c_1, c_5, c_6\}
\end{aligned} \tag{1}$$

The optimal hitting set can be formulated as an integer linear program.

$$min \sum_{i \in \{1..|C|\}} w_i \cdot x_i \tag{2}$$

$$\sum_{i \in \{1..|C|\}} x_i \cdot h_{ij} \geq 1,\ \forall\ j \in \{1..|\mathcal{H}|\} \tag{3}$$

$$x_i = \{0, 1\} \tag{4}$$

- w_i is the input cost/weight associated with clause i in
- x_i is a boolean decision variable if constraint/clause c_i is chosen or not.
- Equation 3, h_{ij} is a boolean input variable corresponding to if constraint/clause i is in set to hit j.

4 Future Work

References