conflict-based-search

nguyenngockhanh.pbc

March 2021

Given a set of resources R and a set of k agents A. Each agent associates with a subset of resources by the cost function $c : A \times P(R) \rightarrow \mathbb{R}$. Our goal is to find such k disjoint subsets of R associate with k agents that minimize the total cost.

In conflict-based search, we define a conflict tree with each node has 3 properties: **constraint**, **assignment** and **cost**. **constraint** is a set of constraints, **assignment** is the least cost function $a : A \rightarrow P(R)$ that uniquely assigns each agent to a subset of resources that satisfies the **constraint** and **cost** is the cost of **assignment**.

The branching rules is defined as follow:

(1) if there is no conflict (all subsets are disjoint), the node is a terminal node.

(2) Choose a conflict (a resource in common of more than one agent), branch-out to $m+1$ child nodes where m is the number of agent taking that resource. In each of the first m child nodes, add a constraint assigning the respective agent to take the resource. In last child node, add a constraint preventing all agents to take the resource.

To elaborate more on the branching rule (2), consider a node with properties as follow:

constraint: { c_1 }
assignment: $a_1 \rightarrow \{r_1, r_2\}, a_2 \rightarrow \{r_2, r_3\}, a_3 \rightarrow \{r_2, r_4\}$
cost: some real number.

Suppose we choose r_2 to branch-out, the first child node is:

constraint: { c_1, (assign r_2 to a_1) }
assignment: least cost assignment satisfies the **constraint**
cost: some real number.

The next two child node is constructed in the same manner by replacing the constraint on a_1 to a_2 and a_3 respectively.

The last child node is:

constraint: { c_1, (do not assign r_2 to any of $\{a_1, a_2, a_3\}$) }
assignment: least cost assignment satisfies the **constraint**
cost: some real number.

By splitting the conflict tree node in this way, there is no duplicate node since $m+1$ child nodes split from a parent node are mutually exclusive. In the case of $m = 2$, the branching rules reduced to the method described in [**?**].

Using best first search on the conflict tree guarantees to find the optimal assignment by the following two arguments:

Lemma 1 (Complete)**.** *Root node (empty* ***constraint****) permits the least cost terminal node if one exists.*

Lemma 2 (Lower bound)**.** *The cost of a conflict tree node is the lower bound of all terminal nodes it permits.*

Lemma 1 can be proved by verifying the conflict tree is limited depth and for every branch-out, the least cost terminal node belongs to either one of the branches.

Lemma 2 can be proved by contradiction since adding more constraints, the cost of a node cannot decrease.