Introduction to Artificial Intelligence Lab

Lab 7: Tabu Search Algorithm

Taxonomy

Tabu Search is a Global Optimization algorithm and a Metaheuristic or Meta-strategy for controlling an embedded heuristic technique. Tabu Search is a parent for a large family of derivative approaches that introduce memory structures in Metaheuristics, such as Reactive Tabu Search and Parallel Tabu Search.

Strategy

The objective for the Tabu Search algorithm is to constrain an embedded heuristic from returning to recently visited areas of the search space, referred to as cycling. The strategy of the approach is to maintain a short term memory of the specific changes of recent moves within the search space and preventing future moves from undoing those changes. Additional intermediate-term memory structures may be introduced to bias moves toward promising areas of the search space, as well as longer-term memory structures that promote a general diversity in the search across the search space.

Procedure

Algorithm (below) provides a pseudocode listing of the Tabu Search algorithm for minimizing a cost function. The listing shows the simple Tabu Search algorithm with short term memory, without intermediate and long term memory management.

```
Input: $TabuList_{size}$
Output: $S_{best}$
$S_{best} \leftarrow$ ConstructInitialSolution()
TabuList $\leftarrow \emptyset$
While ($\neg$ StopCondition())
  CandidateList $\leftarrow \emptyset$
  For ($S_{candidate} \in Sbest_{neighborhood}$)
    If ($\neg$ ContainsAnyFeatures($S_{candidate}$, TabuList))
      CandidateList $\leftarrow S_{candidate}$
    End
  End
  $S_{candidate} \leftarrow$ LocateBestCandidate(CandidateList)
  If (Cost($S_{candidate}$) $\leq$ Cost($S_{best}$))
    $S_{best} \leftarrow S_{candidate}$
    TabuList $\leftarrow$ FeatureDifferences($S_{candidate}$, $S_{best}$)
    While (TabuList $>$ $TabuList_{size}$)
      DeleteFeature(TabuList)
    End
  End
End
Return ($S_{best}$)
```

Figure 1. Pseudocode for Tabu Search.

Heuristics

Tabu search was designed to manage an embedded hill climbing heuristic, although may be adapted to manage any neighborhood exploration heuristic.

 Tabu search was designed for, and has predominately been applied to discrete domains such as combinatorial optimization problems.

 Candidates for neighboring moves can be generated deterministically for the entire neighborhood or the neighborhood can be stochastically sampled to a fixed size, trading off efficiency for accuracy.

 Intermediate-term memory structures can be introduced (complementing the short-term memory) to focus the search on promising areas of the search space (intensification), called aspiration criteria.

 Long-term memory structures can be introduced (complementing the short-term memory) to encourage useful exploration of the broader search space, called diversification. Strategies may include generating solutions with rarely used components and biasing the generation away from the most commonly used solution components.

Exercise

In this week, you implement a Tabu Search algorithm to solve a scheduling problem. Table 1 and 2 describes a scheduling problem. Each entry in Table 1 represents how long it takes for switching from one job to another job. For example, the entry [1,2] ([Job1, Job2]=12) is the time for switching job 1 to job 2 in Table 1. The entries in Table 2 is the time a machine completes the corresponding job. For example, the entry [2,3] ([Machine2, Job3]=5) is the time machine 2 completes job 3.

	Job1	Job2	Job3
Job1	0	12	10
Job2	4	0	8
Job3	6	10	0

Table 1

	Job1	Job2	Job3
Machine1	10	4	8
Machine2	12	9	5

Table 2

The problem in here is to assign each of the 3 jobs to one of the machines in such a way that the total time to complete all jobs is minimized. Implement a simulated annealing algorithm for this problem.