A Simulated Annealing Approach for the Capacitated Minimum Spanning Tree Problem

José Torres-Jiménez
ITESM-Morelos
jtorres@campus.mor.itesm.mx

Raul Pinto-Elías
CENIDET
rpinto@cenidet.edu.mx

Agustín García-Romero
ITESM-Morelos
al374654@academ02.mor.itesm.mx

Abstract

In this paper we present a novel approach to solve an NP-Complete problem that is very important from the theoretical and practical point of view, namely the Capacitated Minimum Spanning Tree problem (CMST). The novel approach has the following features: a) It is based on the simulated annealing (SA) algorithm; b) It represents a tree with $N$ edges with $N-2$ integers; and c) It defines a consistent ordering between feasible and unfeasible trees. Our SA implementation was tested against the most referenced algorithms for the CMST: Essau-Williams algorithm, Prim algorithm, and Kruskal algorithm. The results indicate that our novel approach is very promising to solve CMST problem instances, because it consistently obtains the best results (for the tested cases) but it consumes more time.

1. Introduction

The capacitated minimum spanning tree problem (CMST) has theoretical and practical importance, given that, it involves minimization of a cost and satisfaction of a constraint, it is interesting from the theoretical point of view; in the practical side the CMST appears in the design of teleprocessing networks, hydraulic networks, and automotive roads.

The CMST can be formulated using graphs, we refer to a CMST problem instance as consisting of $G=(V, E, F, C, K)$ where $V$ represents a set of $N$ vertices (numbered from 0 to $N-1$), $E$ indicates a set of edges $(i,j)$ where both $i$ and $j$ belongs to $V$, $F$ refers to a $N$ non-negative flows for the vertex set (the values of $F$ are in the range from 0 to $K$), $C$ defines the non-negative cost for the edge set, and $K$ represents the flow capacity that must not be exceeded by any edge. A special vertex is the concentrator vertex (similar to a sink vertex) to which all the flows go, normally this vertex is the zero and $F_0=0$. Now, the CMST problem can be stated as, given a graph $G$ find $N-1$ edges belonging to $E$ in such a way that: a) the edges represent a tree (connected graph without loops); b) the flow capacity $K$ is not exceeded in any edge; and c) the summation of the cost of the $N-1$ selected edges is a minimum.

The CMST problem is an NP-Complete problem[1] for which a lot of heuristic algorithms has been proposed [2,3,4,12] and many exact algorithms were used too[5]. In one side the heuristic algorithms are very fast but they deliver (in many cases) a local optima solution, in the other side the exact algorithms consume an exponential amount of time but they guarantee to deliver an optimum solution. Between this extremes there are a kind of algorithms that consume more time than the heuristic algorithms but less than the exact algorithms, and the quality of the solutions they deliver is in between the heuristic and the exact algorithms. Examples of this "in-between" algorithms are Genetic Algorithms[6] (GA), Tabu Search algorithms[7,8] (TS) and Simulated Annealing algorithms[9] (SA).

In this paper we present the results that we have obtained using a SA approach to the CMST problem using a special representation and a special energy function.
2. Simulated Annealing

The SA algorithm is based on the analogy between the simulation of the annealing of solids and the solution of combinatorial problems. The physical annealing process refers to the heating of a solid material until it goes to a liquid phase, then a slowly cooling process is conducted until the material goes to a solid phase; in this way the atomic structure of the resulting solid material is of low energy. A high level computer program to accomplish the simulation of the physical annealing is:

**Program Simulated Annealing**

**Begin**

Temperature=High Enough Value

R=Random Configuration;

**Repeat**

**Repeat**

S=Perturb(R)

If Energy(S) <= Energy(R) R=S

Else If exp((Energy(R)-Energy(S))/Temperature) > Random(0,1) R=S

Until equilibrium is reached

Temperature=Temperature*Factor

Until stop criterion

**End**

In order to solve combinatorial problems like the CMST it is very important to define the **Perturb** and **Energy** functions in a correct way, we concentrate next in the definition of these functions.

3 Perturbation Function

Given that a solution to a CMST instance must be a tree, we have to guarantee that a perturbation to a tree must be another tree. In this way we have two main options to define **Perturb**: a) Use an edge representation and verifying that the perturbation produces a connected, no-loops graph; b) Use a special representation that guarantees implicitly that all the perturbations produce a tree. Of these options we selected the second one.

In [10][11] we found that there are \(N^{N-2}\) trees with \(N\) vertices and the fact that there is a bijective mapping involving trees (consisting of \(N-1\) vertices) and sequences of size \(N-2\), we decided to use a sequence \(S\) to represent a tree \(T\) in our SA implementation. The main benefit of using the sequence representation is the fact that all the sequences are valid trees, then the **Perturb** function is extremely simple: Select one of the \(N-2\) integers and assign to it a random vertex number in the range 0 to \(N-1\). Another benefit of using this representation, is that in the process of mapping a sequence \(S\) to a tree \(T\), we obtain both the amount in which a constraint is not accomplished and the cost of the tree.

For space limitations we only give the algorithm **S2T** to map \(S\) to \(T\), in this algorithm an array \(S\) contains the \(N-2\) elements of the sequence, the array \(T\) contains the \(N-1\) edges of the resulting tree, and the array \(D\) initially contains the number of times that a vertex number appears in \(S\).

**Program S2T**

**Begin**

\(D[i]=\)Number of times the vertex i appears in \(S\);

Repeat \(N-3\) Times

Vertex1=The Next Element of \(S\)

Vertex2=The Index of \(D\) that has a zero value

The Next Edge of the \(T\) tree is (Vertex1, Vertex2)
Decrease by 1 D[Vertex1] and D[Vertex2]
End Repeat
Vertex1=The Index of D that has a zero value
Decrease by 1D[Vertex1]
Vertex2=The Index of D that has a zero value
The N-1 Edge of the T tree is (Vertex1, Vertex2)
End

4. Energy Function
All the $N^{N-2}$ trees are a potential solution for a CMST instance of size $N$, but many of these trees may violate the capacity constraint and/or they are not of minimum cost. Many CMST heuristic algorithms only moves through solutions that do not violate the capacity constraint, we believe that maybe this is the main reason of local-optimality of these algorithms. Think for example that there is a solution $W$ of low cost but has a small violation of the constraint, it is very possible that a small change to $W$ gives a zero violation of the constraint and low cost. Taking this in account we decided to permit that SA algorithm moves through all the space solution, and for this objective we constructed an Energy function that includes the cost and the capacity constraint violation. The Energy function that we used has the ability of establishing an ordering among the solutions: any zero violation tree has lower energy than any nonzero violation tree.

The Energy function calculation needs the values: a) MAXC the cost of the maximum spanning tree; b) VIOL the amount of capacity constraint violation of the solution; and c) COST the cost of the solution; then Energy=MAXC*VIOL+COST.

5. Results
In order to test the goodness of our SA algorithm we decided to compare our results with Esau-Williams algorithm[3], Prim Algorithm[4] and Kruskal algorithm[12]. We used three size problems $N=10$, 25 and 50, and also we varied the capacity constraint $K$, the Table 1, presents the results for the four algorithms and the Table 2 indicates the running times for each algorithm. The Figures 1, 2 and 3 present the results for $N=10$, 25 and 50 respectively. The values of $F$ and $C$ for the instances tested can be obtained from the first author.

6. Conclusions
The consistent winner in all the 26 cases that were tested is the SA algorithm, we believe that there are two main reason for the results of our algorithm:
• The representation that we used permitted to our Perturb function a (potential) access to the whole search space.
• The Energy function assign a value to all the solutions, feasible or unfeasible, and this assignment is consistent, i.e. there is no solution that violates the constraint capacity that has a lower value than any feasible solution.

The only disadvantage of the SA algorithm is the amount of time that it requires, currently we are trying to improve the response time in two branches: a) the fine tuning of the SA parameters (specially in the cooling process); and b) the parallel implementation of the SA algorithm in an IBM SP2.

References

<table>
<thead>
<tr>
<th>N</th>
<th>K</th>
<th>Simulated Annealing</th>
<th>Esau-Williams</th>
<th>Prim</th>
<th>Kruskal</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>19</td>
<td>351</td>
<td>363</td>
<td>366</td>
<td>394</td>
</tr>
<tr>
<td>10</td>
<td>24</td>
<td>291</td>
<td>335</td>
<td>376</td>
<td>351</td>
</tr>
<tr>
<td>10</td>
<td>28</td>
<td>238</td>
<td>320</td>
<td>337</td>
<td>301</td>
</tr>
<tr>
<td>10</td>
<td>38</td>
<td>183</td>
<td>239</td>
<td>219</td>
<td>235</td>
</tr>
<tr>
<td>10</td>
<td>43</td>
<td>172</td>
<td>209</td>
<td>220</td>
<td>209</td>
</tr>
<tr>
<td>10</td>
<td>49</td>
<td>172</td>
<td>209</td>
<td>220</td>
<td>209</td>
</tr>
<tr>
<td>10</td>
<td>56</td>
<td>145</td>
<td>165</td>
<td>215</td>
<td>158</td>
</tr>
<tr>
<td>25</td>
<td>25</td>
<td>457</td>
<td>591</td>
<td>610</td>
<td>503</td>
</tr>
<tr>
<td>25</td>
<td>27</td>
<td>406</td>
<td>517</td>
<td>536</td>
<td>503</td>
</tr>
<tr>
<td>25</td>
<td>30</td>
<td>349</td>
<td>538</td>
<td>476</td>
<td>393</td>
</tr>
<tr>
<td>25</td>
<td>39</td>
<td>273</td>
<td>431</td>
<td>314</td>
<td>306</td>
</tr>
<tr>
<td>25</td>
<td>40</td>
<td>230</td>
<td>486</td>
<td>312</td>
<td>316</td>
</tr>
<tr>
<td>25</td>
<td>43</td>
<td>221</td>
<td>393</td>
<td>308</td>
<td>302</td>
</tr>
<tr>
<td>50</td>
<td>477</td>
<td>8171</td>
<td>24073</td>
<td>26014</td>
<td>23905</td>
</tr>
<tr>
<td>50</td>
<td>627</td>
<td>9499</td>
<td>20104</td>
<td>20971</td>
<td>19304</td>
</tr>
<tr>
<td>50</td>
<td>777</td>
<td>8320</td>
<td>17341</td>
<td>17152</td>
<td>16525</td>
</tr>
<tr>
<td>50</td>
<td>927</td>
<td>8221</td>
<td>15112</td>
<td>15676</td>
<td>15001</td>
</tr>
<tr>
<td>50</td>
<td>1077</td>
<td>9983</td>
<td>13439</td>
<td>14063</td>
<td>13284</td>
</tr>
<tr>
<td>50</td>
<td>1227</td>
<td>8377</td>
<td>12653</td>
<td>12456</td>
<td>12315</td>
</tr>
<tr>
<td>50</td>
<td>1377</td>
<td>8886</td>
<td>11517</td>
<td>12177</td>
<td>11126</td>
</tr>
<tr>
<td>50</td>
<td>1527</td>
<td>8705</td>
<td>11105</td>
<td>11283</td>
<td>10860</td>
</tr>
<tr>
<td>50</td>
<td>1677</td>
<td>8018</td>
<td>11034</td>
<td>11977</td>
<td>10501</td>
</tr>
<tr>
<td>50</td>
<td>1827</td>
<td>7967</td>
<td>10807</td>
<td>10459</td>
<td>10443</td>
</tr>
<tr>
<td>50</td>
<td>1977</td>
<td>7504</td>
<td>10372</td>
<td>9932</td>
<td>9878</td>
</tr>
<tr>
<td>50</td>
<td>2127</td>
<td>8302</td>
<td>9474</td>
<td>9142</td>
<td>9343</td>
</tr>
<tr>
<td>50</td>
<td>2277</td>
<td>8012</td>
<td>9238</td>
<td>9378</td>
<td>9111</td>
</tr>
</tbody>
</table>

Table 1: Results of the runs for the compared algorithms

<table>
<thead>
<tr>
<th>N</th>
<th>Simulated Annealing</th>
<th>Esau-Williams</th>
<th>Prim</th>
<th>Kruskal</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>2</td>
<td>01</td>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>25</td>
<td>20</td>
<td>03</td>
<td>0.10</td>
<td>0.10</td>
</tr>
<tr>
<td>50</td>
<td>120</td>
<td>0.60</td>
<td>1.22</td>
<td>0.60</td>
</tr>
</tbody>
</table>

Table 2: Average running times for the compared algorithms
Figure 1 Results for the CMST problem for N=10

Capacity

Figure 2 Results for the CMST problem for N=25

Capacity

Figure 3 Results for the CMST problem for N=50

Capacity