Questions of clarification?

Authors' Observations

- Connectivity algorithm performs the best in both execution time and quality of solution.
- The interleaved versions are MUCH faster (approx 50%) but result in higher costs (approx 15% to 50% higher).
- The difference between interleaved and non-interleaved is greater for the connectivity heuristic.
- Simulated annealing is only about as good as the worst heuristic (non-interleaved Boundary Heuristic).

My Comments on the Paper

The heuristics seem to be very appropriate to the given problem. The authors alternately seem distressed by the fact that the heuristics are specific to this particular problem type and proud of the fact that they are better than a general algorithm such as simulated annealing. It seems logical to me that a specially designed algorithm should perform better than a general algorithm.

The connectivity heuristic really kills simulated annealing and the boundary heuristic.

Neighborhood Function

- Random initial configuration of regions based on given region sizes.
- Facilities are placed on the nodes closest to the centers of each region. (Presumably the lowest cost position)
- At each step two regions are selected for a 'swap' and two nodes are randomly swapped between the regions.
- Per the simulated annealing algorithm, if the new cost is lower, then that configuration is accepted, otherwise is is accepted based on the acceptance probability at that time.

Results

The Boundary and Connectivity heuristics were run in each of their configurations for graph sizes between 100 and 1000 nodes with both uniform and nonuniform region sizes.

Simulated annealing was run for the maximum execution time of the above configurations with each of the three cooling schedules.

The resulting qualities of the solutions were compared as well as execution time to compare the different heuristics.

Interleaved Location and Allocation

To speed up the algorithm, the authors tried interleaving the location and allocation steps. After each particle was fixed, the closest nodes were allocated to it. This decreased the number of combinations to try in each location step by reducing the sample space.

This naturally decreases the quality (increases the cost) of a solution because not as many potential combinations are tried.

Comparison to Simulated Annealing

To quantify the performance of the heuristics, it was compared to the performanance of simulated annealing.

To keep it fair, simulated annealing was run for the maximum time used by either of the two heuristic algorithms in both interleaved and non-interleaved versions tried for a given graph size so that only the quality of the solution needed to be compared.

Cooling Schedules Simulated annealing was tried with three different cooling schedules:

1. *Linear Cooling* $t_{k+1} = t_k - r$
2. *Polynomial Cooling* $t_{k+1} = \frac{t_k}{1+vt_k}$
3. *Exponential Cooling* $t_{k+1} = \lambda t_k$

Boundary Heuristic

The boundary heuristic defines the potential in terms of the particle interactions as well as the interaction (repulsion) of the particles by the wall according to the formula below:

$$\text{Potential} = \frac{1}{2}\sum_{i=1}^{k}\sum_{j\neq i}\frac{m_i m_j}{d(L(i),L(j))} + \gamma\sum_{i=1}^{k}\sum_{j=1}^{4}\frac{m_i}{d(L(i),W_j)}$$

where

- The first term adds up the particle-particle interactions.
- The second term adds up the particle-wall interactions.
- γ is a weighting factor to control the strength of boundary effects.

Notice that a lower potential is achieved by moving the particles apart, especially the 'heavier' particles. Thus, the particles carve out space for their regions by repelling the other particles.

Connectivity Heuristic

Rather than worrying about the boundary, define the potential such that nodes with high degree (connected edges) are favored. The idea being that, in general, a node in the 'center' of a region has the closest access to the most neighboring nodes.

The potential is defined as:

$$\text{Potential} = \frac{1}{2}\sum_{i=1}^{k}\sum_{j=1,j\neq i}^{k}\frac{m_i \cdot m_j}{d(L(i),L(j)) \cdot N(i) \cdot N(j)}$$

NOTE: There appears to be a notational error in the paper which I have corrected.

- δ_{kn} - allocation of nodes to facilities, value is either 1 or 0. Constrained so that a node can only be assigned to one facility and a facility has exactly m_i nodes assigned to it
- $L(k)$ - function that determines the location of the facility. Note that facilities must be located on graph nodes
- $f(L, \delta)$ - cost function for a particular location L and allocation δ

$$f(L,\delta) = \sum_{i=1}^{k}\sum_{j=1}^{n} d(L(i), j)\delta_{ij}$$

Goal

'Find locations for the facilities (L) and allocation of nodes to facilities (δ) to minimize f.'

Simplification and Efficiency

- As is, the problem would result in an unstable solution so the Euclidean plane is discretized so that facilities/particles can only be placed on graph nodes.
- To speed up the program, aggregation and decomposition is used.
  - The facilities are sorted in increasing order
  - All of the particles are combined into one huge particle
  - The smallest particle is removed from the aggregate particle and the lowest potential arrangement is determined by a quadratic search. The position of the smaller particle is fixed and the process is repeated
  - The time required to determine the potential at each step (using the functions on the next few slides) if $O(k)$
  - The search takes $O(kn^2)$
  - The total time is $O(kn)^2$

The Particle Physics Analogy

To find the absolute minimum cost configuration would take too much CPU time so the problem is compared to a physics problem in which:

- k particles with forces between them interact
- The capacities are used as the 'mass' of a facility/particle and the particles push each other around into a minimum energy configuration using the repelling forces

The hope is that the low energy configuration of the particle problem will serve as a low cost solution to the location-allocation problem.

Mathematical Formulation

- $G(V, E)$ - connected Euclidean graph with $|V| = n$ verticies
- $N(u)$ - number of edges that connect to node u
- $d(u, v)$ - length of shortest path from node u to node v
- k - number of facilities
- m_i - number of nodes to be assigned to facility i. $\sum_{i=1}^{k} m_i = n$, ie. all nodes must be assigned to a facility

Capacitated Facility Location

- n nodes on a connected Euclidean graph
- k facilities to be places at a node
- **capacities** $m_1, ..., m_k$ are the number of nodes each facility MUST serve rather than just an upper bound
- cost function is the sum of distances between all of the facilities and their nodes

Allocation

Allocation of nodes to facilities is performed with a standard method from another paper.

1. Nodes select the nearest facility.
2. Regions with not enough nodes steal the closest nodes from regions with too many nodes.

There is no requirement that the regions for each facility be contiguous but it just tends to end up that way because contiguous regions lead to lower cost.

A New Particle Physics-Based Heuristic for a Euclidean Location-Allocation Problem

by Rahul Simha, Weidong Cai, and Valentin Spitkovsky

presentation by Brendan Sibre March 22, 2000

What we are talking about

- The goal of the paper is to present two heuristic algorithms to solve a particular location-allocation problem faster and better than simulated annealing.
- The inspiriation for the algorithms is drawn from particle physics (similar to simulated annealing).
- It focuses on a capacitated facility location problem on Euclidean graphs (2 dimensional).