Second report on the paper "Iterated Tabu Search and Variable Neighborhood Descend for packing unequal circles into a circular container".

About the implemented algorithms

Author: Wilbert Pumacay
July 31, 2017

1 Description of the algorithms implemented

As we described in the first report, the algorithm is divided into 3 phases: diversification, intensification and optimization. The algorithm could be described with the following pseudocode

Algorithm 1 ITS-VND algorithm

1: *Set initial configuration* (R, X)
2: **while** *(Stop criterion)* **do**
3: $(R, X) = DiversificationStep((R, X))$
4: $(R, X) = IntensificationStep((R, X))$
5: $(R, X) = OptimizationStep((R, X))$
return (R, X)

The code structure can be described with Fig.1 .

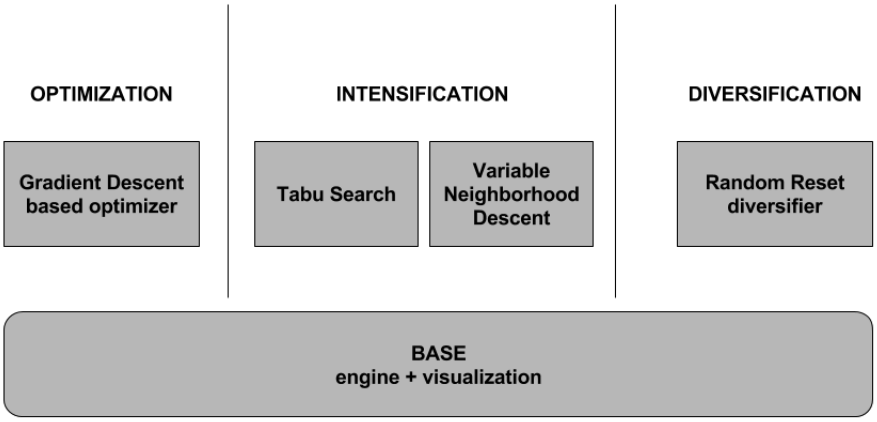

Figure 1: Description of the code structure

As we can see, we have divided the three phases into modules, each one of them in charge of the related phase of the algorithm. All these modules share some functionality, which is enclosed in the BASE module. These modules are described as follows:

1. **Optimization module**
   The optimization module is defined in the *itsvnd/optimization/* folder and implements a gradient descent based optimizer (see the *LGradientDescentOptimizer.h* file). We chose this optimizer instead of the LBFGS optimizer used in the paper. This optimizer is simpler than the LBFGS optimizer, but still yields good results, allowing to reach local minima in very few iterations and returning good configurations.

2. **Intensification module**
   This module implements the metaheuristics of the paper. It is defined in the *itsvnd/intensification/* folder, and implements *Tabu-Search* (*LTSintensifier.h* file) and *Variable Neighborhood Descent* (*LVNDintensifier.h* file). Both metaheuristics use the swap and insert neighborhoods, as opposed to the paper which only uses the insert neighborhood in the VND metaheuristic. This was decided becaused the intensifier yielded better results if both neighborhoods were used in both metaheuristics.

3. **Diversification module**
   This module implements the random-reset strategy described in the paper. This module is defined in the *itsvnd/diversification* folder (*LRRdiversifier.h* file).

4. **Base module**
   This is the core module, which implements the base functionality shared by the other modules, such as the configuration, circle, solver, etc. . This module is defined in the *engine/* folder. There is also a visualization module written in Qt which is part of this module. We use this to visualize how well are our solutions, as we can see in figures 2 and 3.

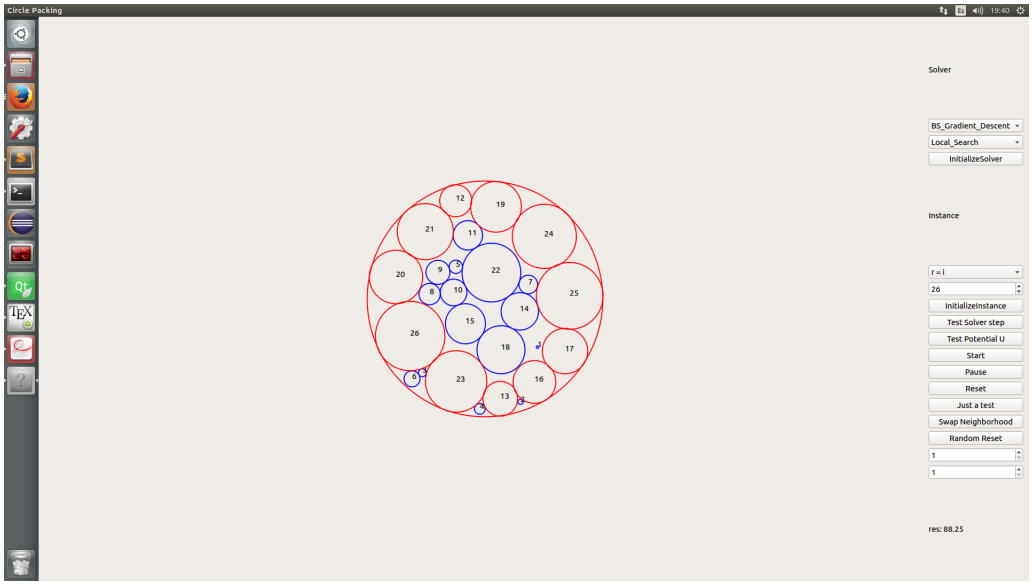

Figure 2: Qt based visualizer

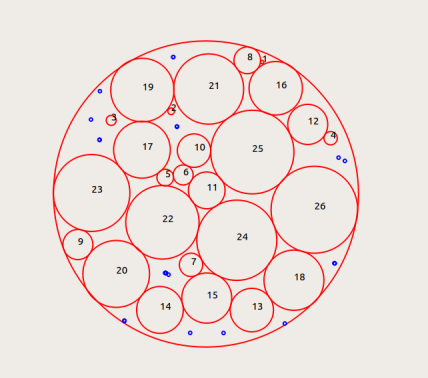

Figure 3: Visualizer and vacant points for the insert neighborhood

This describes all the code implemented for the project. The only part different than the paper is the optimizer, as described earlier.

2 Plan for experimentation

We have use as toy instance the $r = i$ instance. We have tested and the implemented method is off by a 7.5 percent from the results in the paper. We plan check for improvements in the algorithm and test it in the following way.

- Check with the $r = i$ instance for $n = [25 - 50]$ using only the optimizer to see how much can be achieved by optimization alone.
- Check with the $r = i$ instance for $n = [25 - 50]$ using a basic implementation of a local search.
- Check with the $r = i$ instance for $n = [25 - 50]$ using the Variable-Neighborhood-Descent method.
- Check with the $r = i$ instance for $n = [25 - 50]$ using the Tabu-Search method.
- Check with the $r = i$ instance for $n = [25 - 50]$ using the ITS-VND method.
- Check with the $r = i, r = \sqrt{i}, r = \frac{1}{\sqrt{i}}$ instances for $n = [25 - 50]$ using the ITS-VND method.
- Check for improvements in the neighborhoods. Try to use a better neighborhood definition instead of the current swap and insert neighborhoods described in the paper.