HILL CLIMBING

https://scialert.net/fulltext/?doi=jas.2013.919.923

ONE DIMENSIONAL BIN PACKING PROBLEMS

One dimensional packing problem is an NP-hard problem. It consists of a set of pieces that must be packed into the fewest number of bins. This problem has a set of items of a fixed weight and a set of bins of fixed capacity. The packing process must consider these constraints (hard constraint):

- Each item must be assigned to one bin only
- The total weight of items in each bin must be less or equal to the bin capacity

The aim is to minimise to the number of used bins. A mathematical formulation for the fitness of the classical one dimensional bin packing problem is shown in Eq. (1) (Martello and Toth, 1990):

(1)

$$\text{Fitness} = \sum_{i=0}^{n} y_i$$

Subject to:

$$\sum_{j=1}^{n} w_j x_{ij} \ll c y_i, \text{ where } i1N = \{1, \ldots, n\}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \text{ where } j1N$$

$$y_i \in \{0,1\}, \text{ where } i1N,$$

$$x_{ij} \in \{0,1\}, \text{ where } i1N, j \in N$$

where, xij is 1 indicates that piece j is packed into bin i, or otherwise xij is 0; yi is 1 if a bin i contains some pieces, ‘0’ otherwise. n is the number of available bins (and also the number of pieces).

We used five instances of one dimensional packing problem currently available in the 2011 HyFlex software (Table 1). The set of initial solutions are generated as follows:

- Generate a random sequence of items
- Pack them one by one into the first bin into which they will fit “first fit heuristic”
- Evaluate he quality of solution using Eq. 2 (Hyde et al., 2010)

To evaluate the fitness of the solution, this work employs an alternative fitness function to the number of bins as in equation (2) suggested by Hyde et al. (2010):

(2)

$$\text{Fitness} = 1 - \left(\frac{\sum_{i=1}^{n} \left(\frac{\text{fullness}}{C} \right)^2}{n} \right)$$

where, n is the number of bins, fullnessi is the sum of all the pieces in bin i and C is the bin capacity. This will avoid large plateaus in the search space around the best solutions.

LOCAL SEARCH HEURISTICS

This study implemented three basic local search heuristics. These are:

- Hill Climbing/simple descent (HC): The heuristic choose the first improving neighbour that is better than the current solution. Then, an improving neighbour is immediately selected to replace the current solution. The neighbour is randomly evaluated. The heuristic terminate when there is no improved neighbours for certain iteration or when the termination criterion is met

EXPERIMENTAL SETUP AND RESULTS

This work only use a simple pertubation heuristic that is a simple swap, where two different pieces are selected at random and swapped them if there is a space and if it will produce an improvement in fitness

where the best, average and the worst quality solution obtained by HC are better than SA and MSA

———-

https://orca.cf.ac.uk/11334/1/Lewis%2C_R_General_Purpose_Hill_Climbing.pdf

4 Application to the Bin Packing Problem

In this section we now move our attention to the second OIMGP considered in this study – the bin packing problem (see Section 1 for a definition). In the next section we describe our application of the HC method to this problem; Section 4.2 then contains details on the IG and GGA-version used for the comparison. Finally, Section 4.3 gives details on the experiments performed and results gained.
For this problem, recall that n represents the number of items in a problem instance, and χ the smallest number of groups (i.e. bins) that these items can be feasibly packed into. In addition, let s[i] represent the size of an item i, and let C represent the maximum capacity of the bins. Finally, let F(g) represent the "fullness" of a particular bin g in a candidate solution (that is, the total size of all of the items that have been assigned to bin g). Note that none of the algorithms considered in this comparison ever allow a bin to be over-filled in a candidate solution, thus any group g will always have $F(g) \leq C$.

4.1 A Hill-Climbing Algorithm for Bin Packing.

Our application of the hill-climbing method to the bin packing problem follows a similar pattern to the graph colouring application given in Section 3. First, an initial solution is constructed, this time using the First Fit Descending (FFD) heuristic, which involves sorting the n items according to their size (largest first) before applying the greedy algorithm in the usual way.
The improvement scheme used for this application is described in fig. 12 and is based heavily on the "dominance"-based search methods of Martello and Toth [39]. Given two permutations, π and ρ, the aim is to move items between these in such a way that the "fullness" of each group (bin) in π increases, while the number of items within each group in π does not. As a side effect, this operator also moves smaller items into ρ; though this could also be useful in many cases, as smaller items are often more easy to pack than larger items. The procedure thus operates by attempting to swap a pair of items from a bin in π with a pair from a bin in ρ (lines 2-7); then a pair of items from a bin in π with one item

from ρ (lines 8-13); and finally one item from π with one item from ρ (lines 14-19). For our application, when this procedure is invoked, it is applied repeatedly until a complete parse is performed with no change occurring to either permutation. For simplicity's sake, all remaining operational details of this hill-climbing application were kept the same as in the graph colouring experiments.

```
BPP-IMPROVEMENT-PROCEDURE($\pi$, $\rho$, $C$)
(1)   for ($g \leftarrow 1$ to $G(\pi)$)
(2)      foreach (pair of items $i, j$ in group $g$ in $\pi$)
(3)         for ($h \leftarrow 1$ to $G(\rho)$)
(4)            foreach (pair of items $k, l$ in group $h$ in $\rho$)
(5)               $\delta = s[k] + s[l] - s[i] + s[j]$
(6)               if ($\delta > 0$ and $F(g) + \delta \leq C$)
(7)                  Move items $i$ and $j$ into group $h$ in $\rho$ and move items $k$ and $l$ into group
                     $g$ in $\pi$
(8)      foreach (pair of items $i, j$ in group $g$ in $\pi$)
(9)         for ($h \leftarrow 1$ to $G(\rho)$)
(10)           foreach (item $k$ in group $h$ in $\rho$)
(11)              $\delta = s[k] - s[i] + s[j]$
(12)              if ($\delta > 0$ and $F(g) + \delta \leq C$)
(13)                 Move items $i$ and $j$ into group $h$ in $\rho$ and move item $k$ into group $g$ in $\pi$
(14)     foreach (item $i$ in group $g$ in $\pi$)
(15)        for ($h \leftarrow 1$ to $G(\rho)$)
(16)           foreach (item $k$ in group $h$ in $\rho$)
(17)              $\delta = s[k] - s[i]$
(18)              if ($\delta > 0$ and $F(g) + \delta \leq C$)
(19)                 Move items $i$ into group $h$ in $\rho$ and move item $k$ into group $g$ in $\pi$
```

Figure 12: The improvement procedure used in the bin packing application. Here, $G(\pi)$ represents the number of groups that are contained in the permutation π (similarly for ρ).

4.2 Iterated Greedy and GGA Setup (algoritmo genético em grupo)

In this case the IG algorithm was applied using the same experimental conditions as the previous graph colouring experiments, with an initial solution being produced using the FFD heuristic. For the GGA, meanwhile, we used a hybrid version of the algorithm (HGGA), originally proposed by Falkenauer, which was shown to produce excellent results in other works [21, 22]. For the most part, the operators and evolutionary scheme used for HGGA are the same as those used in Erben's GGA for graph colouring (Section 3.2), though in this case, when performing repair during recombination (Step 4, fig. 5), Falkenauer chooses to use FFD heuristic in order to reinsert any unplaced items. In addition to this, the HGGA also uses a local improvement procedure similar in style to our improvement procedure (figure 12), which is intended for locally improving new offspring candidate solutions when they are first constructed. (Falkenauer reports that this twinning of global and local search techniques produces significantly better results than either method individually – similar findings are also reported by Levine and Ducatelle [34] with their hybrid ant colony algorithm.)
One final point about the HGGA is that the fitness function f used to provide evolutionary selection pressure is:

$$f = \frac{\sum_{g=1}^{G} (F(g)/C)^2}{G}$$

where G represents the number of bins being used in the solution. This function is more fine-grained than, say, simply using G as a fitness function, and therefore allows selection pressure to be sustained for longer during a run. One feature of this function is that it favours solutions with “extremist” bins, as fuller bins are more accentuated by the squaring operation. The upshot of this is that a solution that has, say, two bins that are both 90% full will have a lower fitness than a solution that has one bin that is 100% full and one that is 80% full. (Fitness values of 0.81 and 0.82 respectively).

4.3 Experimental Analysis

—— No site ele faz uma analise comparativa do genético em grupo com o hill climbing, talvez seja interessante pro nosso pra depois, mas como é grande deixei só o essencial do código.————