pseudo random floating point values. For the requirements we first generated a service set with uniformly distributed integer bandwidths in the interval $[1, 5]$. Then we determined for each $v \in V$ the size of the corresponding requirement set R_v, again uniformly in the range $[1, 5]$, and selected a random set of this size from the service set. The maximum ensemble size was $M = 10$.

In each test series we varied the parameters n (number of vertices) and r (number of services) as follows: $n \in \{20, 50, 100\}$, $r \in \{10, 20, 50, 100\}$. For each of the twelve resulting parameter combinations we generated 20 random problem instances with the same number of vertices and services – i.e., a total of 240 instances in each series – for which we computed the following figures:

- GFF: number of colors in a smallest-last coloring of a GFF ensemble assignment;
- SFF: number of colors in a smallest-last coloring of an SFF ensemble assignment;
- BEST: best of the two results;
- SUM: lower bound on the clique packing number, using the generalized Carraghan/Pardalos algorithm with the sum estimate of the packing number (cf. Section 6).

Both the GFF and SFF algorithms were invoked using a service order by decreasing sizes. For the GFF assignments we actually used colorings in which the color of a given ensemble is the same for all vertices, i.e., $f(v, B) = f(w, B)\ \forall\ (v, B), (w, B) \in \mathcal{B}$. Such "regular" colorings can be obtained by coloring the "regular ensemble graph" whose vertices are the ensembles $B \in \mathcal{B}_V$ themselves, with edges connecting all pairs of distinct ensembles occuring in the same or interfering areas. We have found that for GFF assignments regular colorings are often better than ordinary colorings, and their computation also tends to take less time, since the regular ensemble graph usually is much smaller than the ordinary ensemble graph.

As it turned out, the running times were rather moderate (typically less than a second per problem instance on a PIII-500MHz Linux PC, even for the largest instances, with the majority of the running time spent in the calculation of the SUM bounds). In the following figures we have plotted the average "performance ratios" GFF/SUM, SFF/SUM and BEST/SUM for the two test series. Thus all plotted values are ≥ 1, and a value of 1 indicates that only optimal solutions were computed for the given parameter combination. The lower labels on the abscissa denote the number of vertices, the upper labels the corresponding number of services.

Figure 3 shows the results for diameter $d = 0.25$, for which the average density of the graphs ($2|E|/|V|(|V| - 1)$) was about 15.6%. The plot shows that for a comparatively small total number of services the GFF algorithm achieves the best (in the beginning even almost optimal) results. However, if we increase the number of services (which, given a constant average size of the requirement sets, will tend to reduce the "circulation" of individual services), then the SFF ensemble assignments eventually do better (which is not