Fiduccia-Mattheyses algorithm. Implementation and modification

Ivan Sladkov, B01-908, MIPT

April 22, 2023

Being an heuristic for solution of a problem of graph partition, Fiduccia-Mattheyses algorithm does not give a completely precise solution, though resulting partition converges to it. Some modifications may be introduced in order to increase convergence speed or precision of partition. In this work original algorithm was implemented and a modification was added to increase speed of algorithm.

1 On algorithm implementation

The original algorithm was implemented with no clustering and with accent on balance. For a formula

$$rW - S_{max} \leq |A| \leq rW + S_{max} \qquad (1)$$

from , which describes soft balancing, coefficient r is taken as 0.5, while maximum divergence between $|A|$ and $|B|$ is $S_{max} = 1$.

Implementation was made in C++. All the containers used in project are based on Standard Template Library. The data structures used are as follows:

- `std::map<int, std::list<int> > gc` – gain bucket
- `std::vector<bool> erased` – shows whether vertex is in bucket or not
- `std::vector<std::pair<int, iterator> > searchSupport` – structure to facilitate and accelerate search

Search through gain buckets is made as $O(1)$.

2 Modification of algorithm

In order to make algorithm converge faster a «cutoff» modification was implemented: if movement of a vertex gives significant cost increase, then further motion is not done. So function `FMPass()` is transformed into:

```
function FMpass
(gain_container, partitionment) :

solution_cost =
partitionment.get cost()

while not all vertices locked {

move = best_feasible_move()

solution_cost-=
gain_container.get gain (move)

if (solution_cost > best_cost+threshold)
then break

gain_container.lock_vertex(
move.vertex())

gain_update (move)

partitionment.apply (move)
}

roll back partitionment
to best seen solution

gain_container.unlock_all()
```

3 Comparison

Result of comparison is shown in table