

A New Multi-level Algorithms for Balanced Partition Problem on Large Scale Directed Graphs

Xian Yue Li

Lanzhou University <https://orcid.org/0000-0002-6311-8888>

Yufei Pang

State Key Laboratory of Aerodynamics

Chenxia Zhao

Lanzhou University

Yang Liu (✉ liuyang@cardc.cn)

Lanzhou University <https://orcid.org/0000-0003-1053-4674>

Qingzhen Dong

Lanzhou University

Research

Keywords: Graph Partition Problem, Large Scale Graphs, Directed Graphs, Multi-level Strategy

Posted Date: March 31st, 2021

DOI: <https://doi.org/10.21203/rs.3.rs-359089/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Version of Record: A version of this preprint was published at Advances in Aerodynamics on August 5th, 2021. See the published version at <https://doi.org/10.1186/s42774-021-00074-x>.

RESEARCH

A New Multi-level Algorithms for Balanced Partition Problem on Large Scale Directed Graphs

Xianyue Li¹, Yufei Pang^{2,3}, Chenxia Zhao¹, Yang Liu^{2,3*} and Qingzhen Dong¹

*Correspondence:

liuyang@cardc.cn

²State Key Laboratory of Aerodynamics, Mianyang, Sichuan, 621000, China

³Computational Aerodynamics Institute, China Aerodynamics Research and Development Center, Mianyang, Sichuan, 621000, China

Full list of author information is available at the end of the article

Abstract

Graph partition is a classical combinatorial optimization and graph theory problem, and it has a lot of applications, such as scientific computing, VLSI design and clustering etc. In this paper, we study the partition problem on large scale directed graphs under a new objective function, a new case of graph partition problem. We firstly propose the modeling of this problem, then design an algorithm based on multi-level strategy and recursive partition method, and finally do a lot of simulation experiments. The experimental results verify the stability of our algorithm and show that our algorithm has the same good performance as METIS. In addition, our algorithm is better than METIS on unbalanced ratio.

Keywords: Graph Partition Problem; Large Scale Graphs; Directed Graphs; Multi-level Strategy

1 Introduction

Graph partition is a classical combinatorial optimization and graph theory problem. Given a graph G and a parameter k , the aim of this problem is to divide the vertex set of G into k parts, and to optimize the given objective functions. If we restrict the number (or total weights) of vertices of all parts are same or as same as possible, this problem is called balanced graph partitioning problem (BGP). BGP is a standard special case of graph partition problem, and it has a lot of applications, such as scientific computing, VLSI and chips design, image processing and clustering etc. Andreev and H. Räcke [1] showed that BGP is NP-hard even for 2-partition, and there is no constant approximation algorithm. In particular, BPG doesn't admit constant approximation algorithm unless $NP = P$, even for trees and grids [2]. In addition, other cases of graph partition problem with application background have also received extensive attention from researchers, such as hyper-graph partition problem [3, 4], balanced connected graph partition problem [5, 6], and path-partition problem [7], etc. Recently, Buluç et al. [8] surveyed the algorithms design and applications of graph partition problem. Although there is no constant approximation algorithm for BGP, due to its wide applications, many heuristic algorithms had been developed to solve it. Firstly, by using local search strategy, Kernighan and Lin [9] presented an efficient heuristic algorithm for 2-BGP with time complexity $O(n^2 \log n)$. Then, Fiduccia and Mattheyses [10] developed a linear heuristic algorithm. Spectral method [11] is also an important method to solve

BGP. This method divides the given graph into two parts, by using their eigenvalues and eigenvectors of its adjacency matrix or Laplacian matrix. At present, there are many graph partition algorithms based on spectral method [12, 13], which can solve 2-BGP or general k -BGP iteratively.

On the other hand, with the increasing of the problem scale and improvement of the computing power, the size of the graph to be divided is becoming larger and larger, and the number of vertices of the graph can reach 100,000,000 or more. Thus, it is impractical to use the previous algorithms to solve large scale graph partition problem. Based on this, researchers proposed multi-level method and streaming algorithms to solve this problem. The main idea of multi-level method is to convert the original graph into a small scale resulting graph by multiple contraction firstly, then divide the new graph into k -parts, and finally back map and modify the partition of the contracted graph become a partition of the original graph. The popular software and software package of graph partition, METIS [14] and KaHIP [15] are designed based on this method. The main idea of the streaming algorithm is to assign each vertex of the graph into the suitable part one by one, through a specific potential function. The advantage of streaming algorithm is fast and saving memory, and it is very suitable for large-scale graph partition problem. The graph partition software FENNEL is based on streaming algorithm [16].

Although a lot of theoretical results and algorithms on graph partition have been obtained, there are still some problems that have not been explored. The first problem is partition on directed graph. Most of the previous works are on undirected graphs, but for some practical applications, such as multi-subject coupling problem, the corresponding models should be directed graph. Therefore, it is necessary to study the partition on directed graphs. The second one is about the objective function. In the past, researchers often considered the vertex-weight and the edge-weight separately, that is, to optimize some edge-weight objective functions under some vertex-weight constraints. There are few works on objective functions combining the two weights together. Based on this two point, we study the directed graph problem with combined weight function.

The organization of this paper. Some basic conception of graph theory and the mathematical modeling of this problem will be presented in Sect. 2. In Sect. 3, we introduce the main idea and process of our algorithm. The experimental results are exhibited in Sect. 4. In detail, we will verify the stability of our algorithm, determine the some parameters and compare our algorithm with METIS. Finally, the conclusion and future work are given in Sect. 5.

2 Basic Conceptions and Mathematical Modeling

In this paper, we will introduce some conceptions in graph theory and develop the mathematical programming for the new balanced graph partition problem.

A (undirected) *graph* G is an ordered pair $(V(G), E(G))$ consisting of a set $V(G)$ of *vertices*, and a set $E(G)$ of *edges*. Each edge of G is an unordered pair of vertices. If an edge e joins vertices u and v , then u and v are called the *ends* of e . A *directed graph* D is an ordered pair $(V(D), A(D))$ consisting of a set $V(D)$ of *vertices*, and a set $A(D)$ of *arcs* (directed edges). Each arc of D is an ordered pair of vertices. If an arc a joins vertices u to v , then u is the *tail* of a , v is the *head* of a , and u

and v are the *ends* of a . For any graph, if we see each edge $e = uv$ as two arcs (u, v) and (v, u) , then this graph becomes a directed graph. Thus, undirected graphs can be considered as a special class of directed graphs. For any vertex v in D , the notation $A_D^-(\{v\})$ is the sets of arcs whose heads are v , and the notation $A_D^+(\{v\})$ is the sets of arcs whose tails are v . Furthermore, for any vertex subset X , $A_D^-(X)$ ($A_D^+(X)$) is the sets of arcs whose heads (tails) are in X , but tails (heads) are not in X . A set M of independent arcs (no common ends) in a digraph D is called a *matching*. Given a matching M of D , a vertex v is called *matched* (by M) if v is an end of some arc of M ; otherwise, v is called *unmatched*. A matching M of G is *maximal* if for any arc a not in M , $M \cup a$ is not a matching of D .

Given a directed graph $D = (V, A)$ with a weighted function w on $V \cup A$, a k -partition P is a decomposition (V_1, V_2, \dots, V_k) on vertex set V , such that $V_i \neq \emptyset$, $V_i \cap V_j = \emptyset$ for any $1 \leq i < j \leq k$ and $V_1 \cup V_2 \cup \dots \cup V_k = V$. Given a specific k -partition P , for any part j , we define its load

$$L_j^P = w(V_j) + w(A_D^-(V_j)),$$

where $w(V_j) = \sum_{v \in V_j} w(v)$ and $w(A_D^-(V_j)) = \sum_{a \in A_D^-(V_j)} w(a)$. Let L_M^P and L_m^P be the maximum load and minimum load among all parts in P , that is,

$$L_M^P = \max_{1 \leq j \leq n} L_j^P \quad \text{and} \quad L_m^P = \min_{1 \leq j \leq n} L_j^P.$$

Thus, we model the balanced graph partition problem as the following unconstrained two-objective programming,

$$\min_{P \in \mathcal{P}} \rho^P = L_M^P / L_m^P - 1 \quad (1)$$

$$\min_{P \in \mathcal{P}} L_M^P \quad (2)$$

where \mathcal{P} is the set of all k -partitions of G and ρ^P is the unbalanced ratio of the partition P .

As mentioned in Sect. 1, our problem differs from METIS in two points. The first is that METIS can only deal with undirected graphs, but our problem is defined on the directed graphs. The second is the different objections. The mathematical model of METIS as follows,

$$\min \sum_{e \in E_C} w(e) \quad (3)$$

$$s.t. \quad w(V_j) \leq \rho \cdot W(V)/k \quad j = 1, 2, \dots, k; \quad (4)$$

where E_C is the set of edges whose ends are in distinct parts, and $\rho \geq 1$ is the unbalanced ratio of the vertex weights. That is to say, the model of METIS considers vertices and edges separately, but we consider them together.

3 Algorithm

Since the scale of the graphs we're going to deal with are very large (up to 100,000,000 vertices), and the number of parts are also large (up to 100,000), our

algorithm is designed by combining classical multi-level method and recursive partition method.

3.1 Multi-level Stage

Recently, the popular method to partition the large scale graph is multi-level. The main idea of the multi-level method has three phases: iterative contraction, initial partition and modification, and backward mapping. We will introduce the detail of each phase in the following.

PHASE 1: Iterative Contraction. In this phase, we will construct a sequence of directed graphs (D_0, D_1, \dots, D_m) and make the numbers of vertices of graphs to be decreased one by one, where D_0 is the original directed graph. To do this, as the previous strategy, for any current graph D_i , we computer a maximal matching M_i and contract it to obtain the next graph D_{i+1} . This phase is end when the one of the following cases occurs: (i) the number of vertices of current graph is less than ck , where k is the number of parts of the partition and $c = 90$ is the contracted parameter chose by our experiments in next section; (ii) the ratio of contraction $|V(D_{i+1})|/|V(D_i)|$ is large than 80%, that is $|M_i| \leq 20\%|V(D_i)|$. To computer the maximal matching, we will use the following two random methods.

Random Maximum Weight Matching (RMWM). This classical method is used in METIS [14] and other multi-level algorithms [15]. The process of RMWM as follows. The vertices of the graph are chosen by a random order. For a chosen vertex u , if u is already matched by other vertex or its in-neighbors are all matched, we choose the next vertex. Otherwise, u is matched with its unmatched in-neighbor v with the maximum weight of arc (v, u) , that is,

$$v = \arg \max \{w(v, u) \mid v \text{ is an unmatched in-neighbor of } u\}.$$

When all vertices are chosen, we can obtain a maximal matching.

Random Maximum Ratio Matching (RMRM). The motivation to use this matching is the new objective functions. The difference between processes of RMRM and is only when u has unmatched in-neighbors. Since the objective function considers the weights of vertices and arcs together, u is matched with its unmatched in-neighbor v with the maximum ratio of arc-weight to vertex-weight, that is,

$$v = \arg \max \left\{ \frac{w(v, u)}{w(v)} \mid v \text{ is an unmatched in-neighbor of } u \right\}.$$

PHASE 2: Initial Partition and Modification. After iterative contraction, the final graph D_m has at most ck vertices. Thus, we can obtain a good initial partition by greedy strategy fast. In detail, we will use the best fit decreasing (BFD) algorithm similar to that of solving the bin-packing problem. Firstly, we set every part $P_j = \emptyset$ for any $j = 1, 2, \dots, k$ and reordering the vertices with decreasing vertex-weight. For each stage, if we put the current vertex v into the part j , then the load of the part j will become

$$L'_j = L_j + w(v) + \sum_{u \in N^-(v) \cap P_i} w(u, v);$$

and the load of other part i will become

$$L'_i = L_i + \sum_{u \in N^+(v) \cap P_i} w(v, u).$$

Thus, we put the v into the part which can make the maximum load minimum. When the all vertices are visited, the initial partition P is obtained.

The aim of modification is to make the initial partition to be local optimum. The main strategy is local search, that is, move a vertex of the maximum load part into another part to reduce the maximum load, iteratively. In detail, for current iteration, we firstly choose a part P_j with the maximum load. Then, for any vertex v in P_j , we calculate its in-arc-weight $w_i^-(v)$ and out-arc-weight $w_i^+(v)$ with respect to each part P_i ($1 \leq i \leq k$) as follows,

$$w_u^-(v) = \sum_{u \in N^-(v) \cap P_i} w(u, v), \quad w_i^+(v) = \sum_{u \in N^+(v) \cap P_i} w(v, u).$$

Now, if we move vertex v from part P_j into part P_i , then the load of part other than P_i and P_j has not changed, and the new loads L'_j and L'_i become

$$L'_j = L_j - w(v) - w_i^-(v) + w_j^+(v), \quad L'_i = L_i + w(v) - w_i^+(v) + w_j^-(v).$$

For every pair (v, P_i) , we can calculate the maximum load and the sum of loads of the swapped partition. If there exists some swapped partitions whose maximum load less than that of the current partition, then we choose the partition with minimum maximum load instead of the current one, and repeat this operation. Otherwise, if there are some swapped partitions whose maximum load equal to that of the current partition, but the sum of loads are less than that of the current partition, we choose the partition with minimum sum of loads instead of the current one, and repeat this operation; else, the current partition achieves local optimum, and the process of modification is finished.

PHASE 3: Back Mapping. In this phase, it should be mapping the partition of D_{i+1} back to that of D_i , and modify the partition of D_i to be local optimum in each level $i = m - 1, m - 2, \dots, 0$. But since the scale of the original graph is huge and the number of parts is also large, in order to save the memory and speed up the running time, we mapping the partition of D_m back to the partition of D_0 directly.

3.2 Recursive Partition Stage

As the above subsection, the phase of iterative contraction ends when the number of vertices of contracted graph D_m is less than $90k$, where k is the number of parts of desired partition. This implies that if k is large, the scale of D_m is also large and it can make bad performance and long running time. Thus, we use the recursive partition strategy to avoid this.

The main idea of recursive partition method as follows. At the beginning, we divide k into several little numbers, that is, $k = k_1 k_2 \cdots k_t$, where each k_i is a little number (may be ≤ 20). In the first step, we firstly use the multi-level method to obtain a k_1 -partition P of the original graph. Since k_1 is small, we can guarantee

the good performance and short running time. Then, based on the partition P , the whole graph will be decomposed into k_1 subgraphs, and each subgraph is induced by each part in P . Note that the weight of arcs in the subgraphs is the same as that in the original graph, but the weight of every vertex v needs to be changed as follows,

$$w'(v) = w(v) + \sum_{u \in N^-(v) \ \& \ P[u] \neq P[v]} w(u, v),$$

where $P[v]$ is the part which v belongs to P . The purpose of changing vertex-weight is to ensure that the objective value is the same either the partition for whole graph or for the subgraphs. In the second step, we will divide every subgraph into k_2 parts, and obtain $k_1 k_2$ new subgraphs by decomposing all old subgraphs. Hence, in the last step, we have $k_1 k_2 \cdots k_{t-1}$ subgraphs and obtain a k_t -partition of every subgraph. That is, we obtain a partition of the original graph with $k_1 k_2 \cdots k_t = k$ parts.

How to choose recursive partition strategy? Based on our experiments in next section, we find that there is little difference between different strategies. Thus, if k is a power of some integer $b \leq 20$, that is $k = b^t$, then we divide k into $b \times b \times \cdots \times b$.

4 Experimental Results

If we do this, we can obtain a better result on unbalanced ratio and maximum load. Since in many practical applications, each small part is often required to induced a connected subgraph, In this section, our experiment is mainly divided into two parts: design of algorithm and comparison with other algorithms. In the part of design of algorithm, we will test the performance of the two random matching methods, verify the stability of random method, and determine the contracted parameter c and strategy of recursive partition. In the comparison part, we will compare our algorithm with the k -way partition algorithm in METIS on unbalanced ratio, maximum load and running time to evaluate the performance of our algorithm.

The directed graphs used in the experiment consists of two classes, theoretical and practical models. We use the grid graph as the representative of the theoretical model, which can also be regarded as the inner dual graph of the square grid of plane. We consider grid graphs of three sizes Grid-1 with 1,000,000 vertices and 3,996,000 arcs, Grid-2 with 10,890,000 vertices and 43,546,800 arcs, and Grid-3 with 100,000,000 vertices and 399,600,000 arcs, each of which has the random vertex-weight of 120-150, the weight of every arc is about $1/20$ of the weight of its end. In practical models, we use 8 graphs from 3D finite element meshes, two of them from the METIS and others from the real examples. The characters of all graphs are showed in the Tab. 1.

All the experiments were performed on a Dell T7610 graphics workstation with Intel Xeon 2.6GHz CPU (6 cores) and 1866mhz DDR3 32 GB memory.

4.1 Matching Comparison

The aim of the subsection is to test the performance of the two matching contraction methods, and RMRM mentioned in Subsec. 3.1. We do the experiment on five graphs, Grid-1, Grid-2, MDual, FEM-1 and FEM-3. The small-scale graphs (Grid-1, MDual and FEM-1) and large-scale graphs (Grid-2 and FEM-3) are partitioned

Table 1 The Characters of Graphs

| Name of Graph | No. of Vertices | No. of Arcs | Description |
|---------------|-----------------|-------------|--|
| Grid-1 | 1,000,000 | 3,996,000 | Theoretical Grid Graph |
| Grid-2 | 10,890,000 | 43,546,800 | Theoretical Grid Graph |
| Grid-3 | 100,000,000 | 399,600,000 | Theoretical Grid Graph |
| Copter | 55,476 | 704,476 | 3D Finite Element Mesh From METIS |
| MDual | 258,569 | 1,026,264 | 3D Finite Element Mesh From METIS |
| FEM-1 | 122,549 | 735,414 | 3D Finite Element Mesh From Real Example |
| FEM-2 | 3,209,941 | 12,747,458 | 3D Finite Element Mesh From Real Example |
| FEM-3 | 3,899,018 | 15,476,406 | 3D Finite Element Mesh From Real Example |
| FEM-4 | 5,279,053 | 20,998,604 | 3D Finite Element Mesh From Real Example |
| FEM-5 | 6,636,730 | 26,412,712 | 3D Finite Element Mesh From Real Example |
| FEM-6 | 12,982,941 | 51,722,612 | 3D Finite Element Mesh From Real Example |

into 100 and 1000 parts and 1000 and 10000 parts, respectively, where the contracted parameter $c = 90$ and the recursive partition strategies are 10^2 , 10^3 and 10^4 . Because of the randomness of the algorithm, we do each partition 10 times, and then compare the average and maximum values of the unbalanced rate ρ and the max-load L_M . The experimental and comparative results can be seen in following table and figures.

Fig. 1 illustrates that the unbalanced ratios of are better than that of RMRM, except for the maximum unbalanced ratio of 100-partition on MDual. Fig. 2 implies that in term of max-load, while the performance of is better than RMRM, the gap is very small and the maximum ratio is less than 1.012. Hence, we use the method in the following.

Table 2 The Experimental Result on Different Matchings

| Graph | Number of Parts | Type of Matching | Unbalanced Ratio | | Max Load | |
|--------|-----------------|------------------|------------------|---------|----------|---------|
| | | | Average | Maximum | Average | Maximum |
| Grid-1 | 100 | RMRM | 1.05% | 1.36% | 1399118 | 1400935 |
| | | RMWM | 0.36% | 0.50% | 1386662 | 1387988 |
| | 1000 | RMRM | 1.85% | 2.25% | 145745 | 146006 |
| | | RMWM | 0.99% | 1.27% | 145043 | 145334 |
| Grid-2 | 1000 | RMRM | 1.53% | 1.74% | 1527649 | 1531943 |
| | | RMWM | 0.75% | 0.93% | 1511836 | 1514495 |
| | 10000 | RMRM | 2.47% | 2.72% | 159246 | 159435 |
| | | RMWM | 1.34% | 1.68% | 158101 | 158297 |
| MDual | 100 | RMRM | 0.89% | 0.98% | 380119 | 380562 |
| | | RMWM | 0.67% | 1.07% | 379111 | 379689 |
| | 1000 | RMRM | 1.71% | 2.07% | 41004 | 41056 |
| | | RMWM | 1.49% | 1.89% | 40911 | 41028 |
| FEM-1 | 100 | RMRM | 0.89% | 1.20% | 760856 | 761381 |
| | | RMWM | 0.27% | 0.40% | 758603 | 758951 |
| | 1000 | RMRM | 1.63% | 2.08% | 79113 | 79222 |
| | | RMWM | 1.22% | 1.41% | 78978 | 79050 |
| FEM-3 | 1000 | RMRM | 1.28% | 1.37% | 568638 | 568961 |
| | | RMWM | 0.73% | 1.01% | 567277 | 568502 |
| | 10000 | RMRM | 2.11% | 2.30% | 60369 | 60473 |
| | | RMWM | 1.60% | 2.07% | 60203 | 60358 |

4.2 Stability Verification

In this subsection, we will test the stability of algorithm, that is, determining whether will bring a large deviation to the output result. The same graphs with same parts are used in the experiment. we compare the experiment results from

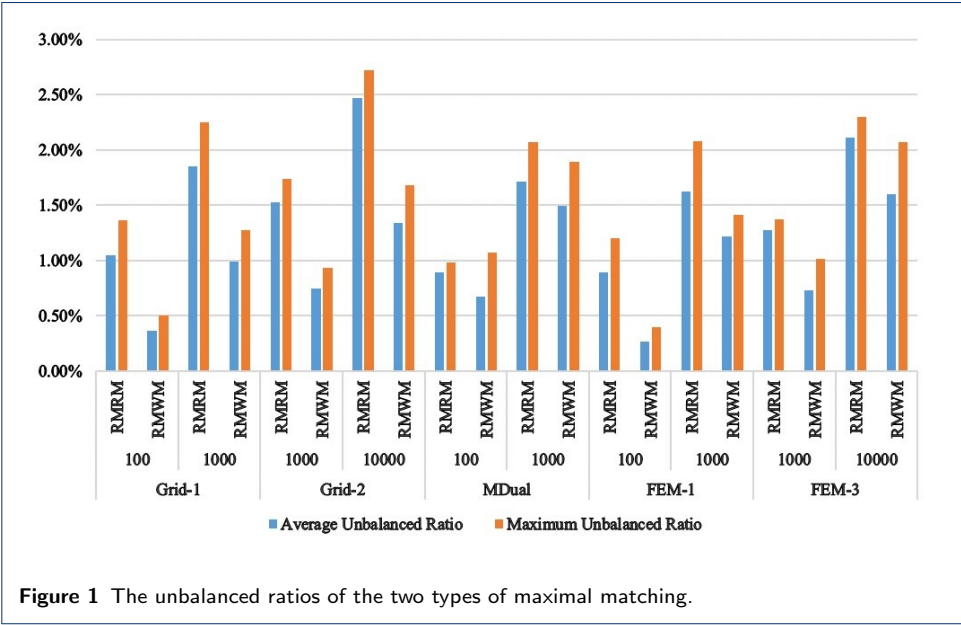


Figure 1 The unbalanced ratios of the two types of maximal matching.

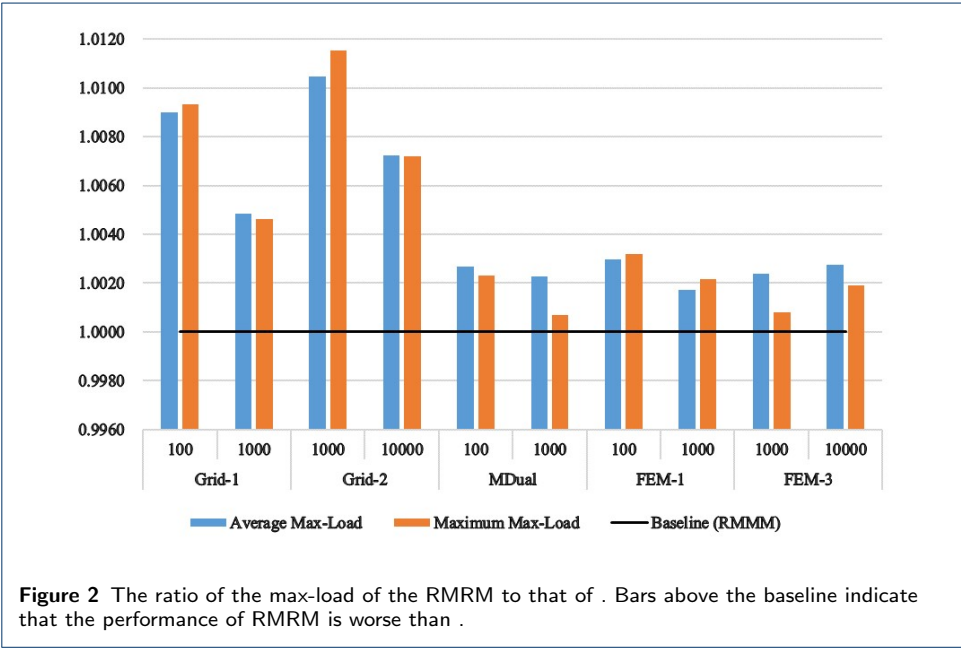
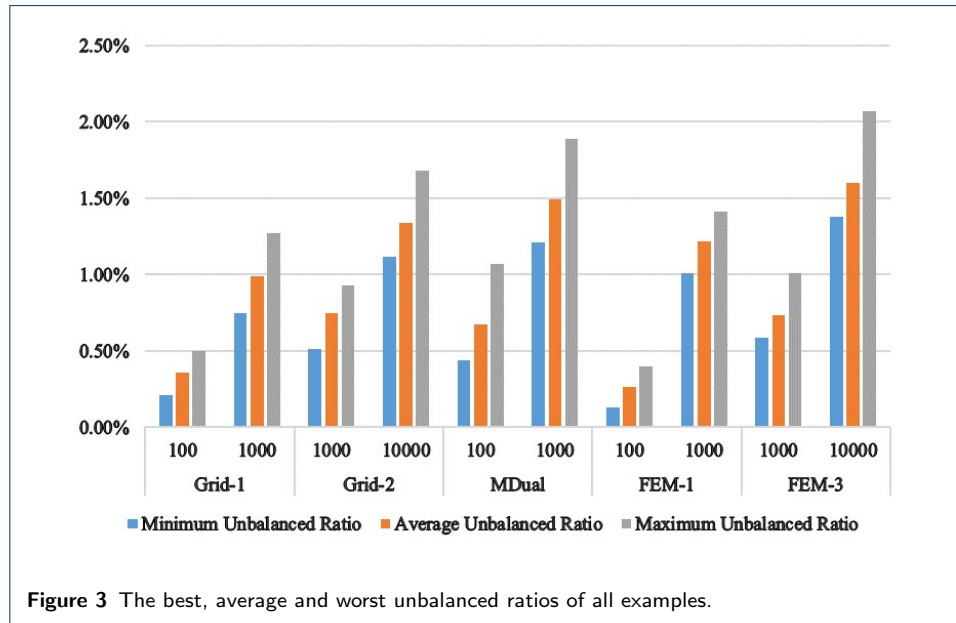


Figure 2 The ratio of the max-load of the RMRM to that of RMMM. Bars above the baseline indicate that the performance of RMRM is worse than RMMM.

unbalance ratio, max-load and running time three aspects, and the detail can be seen in Tab. 3.

Table 3 The Experimental Result on Randomized Stability

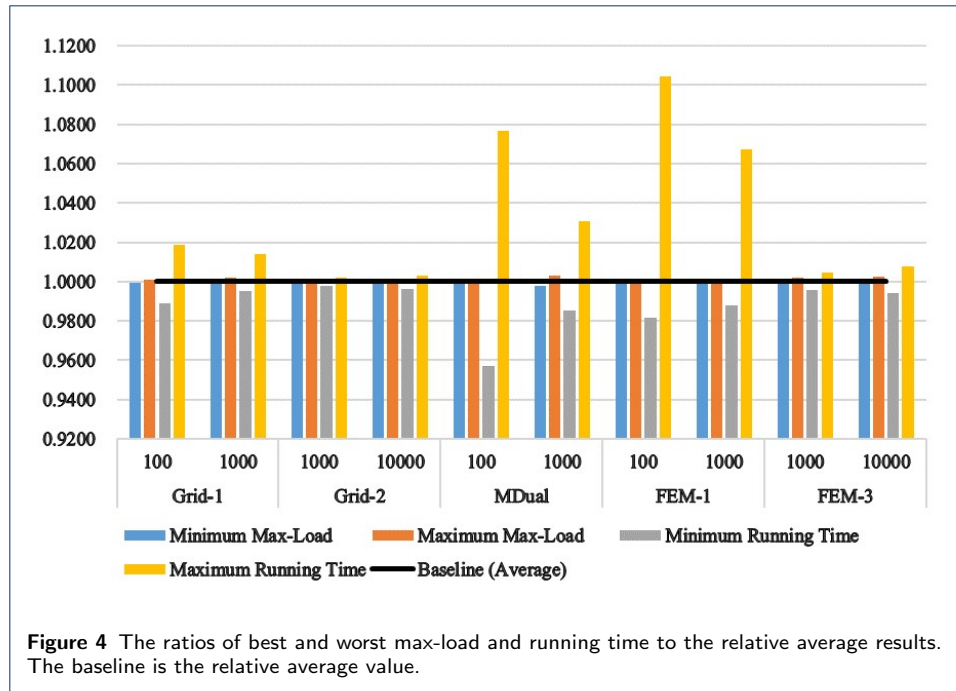
| Graph | Number of Parts | Unbalanced Ratio | | | Max Load | | | Running Time (s) | | |
|--------|-----------------|------------------|---------|-------|----------|---------|---------|------------------|---------|---------|
| | | Best | Average | Worst | Best | Average | Worst | Fastest | Average | Slowest |
| Grid-1 | 100 | 0.21% | 0.36% | 0.50% | 1385641 | 1386662 | 1387988 | 2.000 | 2.022 | 2.060 |
| | 1000 | 0.75% | 0.99% | 1.27% | 144901 | 145043 | 145334 | 2.590 | 2.603 | 2.640 |
| Grid-2 | 1000 | 0.51% | 0.75% | 0.93% | 1497731 | 1511836 | 1514495 | 30.700 | 30.764 | 30.820 |
| | 10000 | 1.12% | 1.34% | 1.68% | 155626 | 158101 | 158297 | 36.880 | 37.009 | 37.120 |
| MDual | 100 | 0.44% | 0.67% | 1.07% | 378740 | 379111 | 379689 | 0.560 | 0.585 | 0.630 |
| | 1000 | 1.21% | 1.49% | 1.89% | 40825 | 40911 | 41028 | 0.860 | 0.873 | 0.900 |
| FEM-1 | 100 | 0.13% | 0.27% | 0.40% | 758180 | 758603 | 758951 | 0.320 | 0.326 | 0.360 |
| | 1000 | 1.01% | 1.22% | 1.41% | 78916 | 78978 | 79050 | 0.500 | 0.506 | 0.540 |
| FEM-3 | 1000 | 0.59% | 0.73% | 1.01% | 561436 | 567277 | 568502 | 13.840 | 13.896 | 13.960 |
| | 10000 | 1.38% | 1.60% | 2.07% | 59135 | 60203 | 60358 | 17.140 | 17.240 | 17.370 |



From Fig. 3, we can see that although there is a gap between the best and the worst, the gap is small, and the biggest is only 0.70%. Furthermore, all of unbalance ratios are quite small, less than 2.00% except the worst result of 10000-partition on FEM-3. Fig. 4 illustrates the max-load and the running time, where the baseline is average values. For each example, the worst max-load is almost equal to the best one; the difference of running time is also very small, and the maximum ratio is about 1.10. Hence, the randomness of algorithm does not bring much deviation, our algorithm is very stable.

4.3 Determining Parameters

In our algorithm, there is a parameter and a strategy that need to be determined. Firstly, we determine the parameter, contracted parameter c mentioned in Subsec. 3.1, by comparing the results with $c = 50, 70, 90, 110, 130, 150$. The experiment was conducted by three representative graphs, Grid-2, MDual and FEM-3, with



the same partitions and same recursive partition strategies as Subsec. 4.1. The comparison results are illustrated in the following three figures.

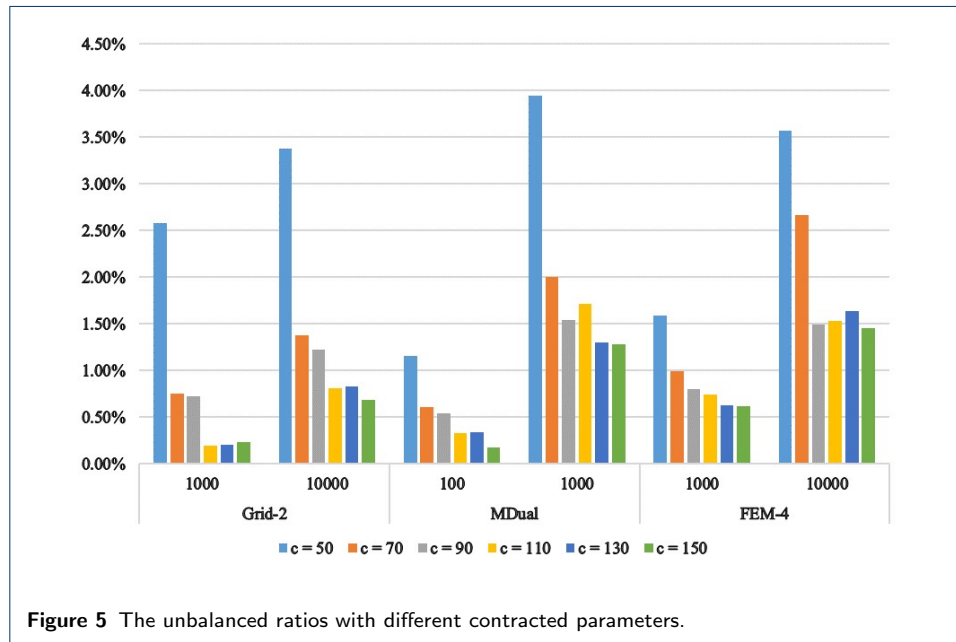


Fig. 5 shows the unbalanced ratios with different contracted parameter c . Fig. 6 and Fig. 7 exhibit the ratios of results of other parameters at maximum load and running time to results of $c = 90$, respectively. From these figures, we can see that the unbalanced ratio will basically decrease with the increase of the contracted parameters, on the contrary, the max-load and the running time will often rise with

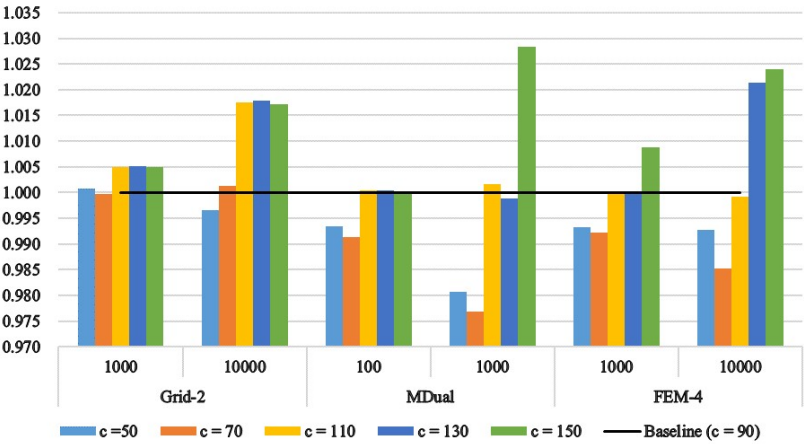


Figure 6 The ratios of max-load of other contracted parameters to that of parameter 90.

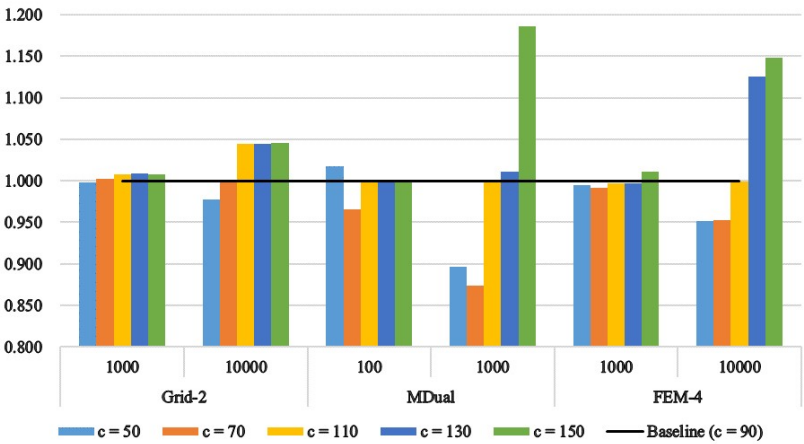


Figure 7 The ratios of running time of other contracted parameters to that of parameter 90.

the increase of the parameters. Overall, good performance is occurred when the parameter is selected as 70, 90, 110. Thus, we will choose the parameter $c = 90$.

For the recursive partition strategy, by dividing the number k and doing corresponding experiments, we find that there is little difference between these results. The deviations of unbalanced ratio and ratio of max-load are at most 0.5% and 0.2%, respectively. Hence, we choose the simplest strategy, that is, divide k into a power of some integer $b \leq 20$. For example, if $k = 1000$, our algorithm is divided into three stages, and each stage does 10-partition.

4.4 Comparison with METIS

In this subsection, we will compare the performance of our algorithm (Graph_Partition) with the k -way partition in METIS by carrying out the experiments on the 11 graphs of Tab. 1. Since METIS can only deal with undirected graphs, we transform each directed graph in Tab. 1 into an undirected graph, by modifying the weight of every edge uv as $\frac{w(u,v)+w(v,u)}{2}$. Then, the resulting undirected graphs are partitioned by the k -way partition. Finally, we calculate the unbalanced ratio and max-load of each graph with respect to the partition. The experimental results can be seen in Tab. 4, and the comparison can be seen in the following figures. Note that since the graph Grid-3 is huge (100,000,000 vertices and 399,600,000 arcs), METIS does not calculate a feasible result.

Table 4 The Experimental Results of Graph_Partition and k -Way Partition

| Graph | Number of Parts | Unbalanced Ratio | | Max Load | | Running Time (s) | |
|--------|-----------------|------------------|--------|----------|---------|------------------|-------|
| | | Graph_P | METIS | Graph_P | METIS | Graph_P | METIS |
| Grid-1 | 100 | 0.28% | 6.18% | 1385877 | 1395467 | 2.03 | 1.18 |
| | 1000 | 0.90% | 6.47% | 144877 | 140671 | 2.60 | 1.92 |
| Grid-2 | 1000 | 0.72% | 6.25% | 1512586 | 1520553 | 30.80 | 10.76 |
| | 10000 | 1.25% | 22.69% | 157978 | 166845 | 36.91 | 29.72 |
| Grid-3 | 10000 | 1.34% | — | 1394690 | — | 422.31 | — |
| | 100000 | 2.00% | — | 145606 | — | 480.40 | — |
| Copter | 100 | 1.88% | 15.03% | 115978 | 90908 | 0.26 | 0.19 |
| | 1000 | 7.07% | 29.37% | 14316 | 10861 | 0.42 | 1.10 |
| MDual | 100 | 0.68% | 7.26% | 378894 | 367047 | 0.57 | 0.35 |
| | 1000 | 1.52% | 8.14% | 40891 | 37684 | 0.87 | 1.21 |
| FEM-1 | 100 | 0.18% | 6.09% | 758264 | 766101 | 0.32 | 1.02 |
| | 1000 | 1.06% | 6.48% | 78941 | 77174 | 0.50 | 1.49 |
| FEM-2 | 1000 | 0.81% | 6.99% | 467329 | 455296 | 11.05 | 7.91 |
| | 10000 | 1.69% | 9.01% | 49573 | 46728 | 13.64 | 19.74 |
| FEM-3 | 1000 | 0.55% | 7.03% | 566882 | 552346 | 13.86 | 8.73 |
| | 10000 | 1.55% | 8.78% | 60179 | 56684 | 17.22 | 22.62 |
| FEM-4 | 1000 | 0.72% | 6.98% | 762047 | 746764 | 19.44 | 11.72 |
| | 10000 | 1.94% | 8.11% | 80081 | 76280 | 23.15 | 26.62 |
| FEM-5 | 1000 | 0.71% | 6.87% | 957775 | 938615 | 25.22 | 14.77 |
| | 10000 | 1.40% | 8.05% | 100549 | 95664 | 30.01 | 31.82 |
| FEM-6 | 1000 | 1.12% | 6.78% | 1857835 | 1828265 | 55.19 | 25.87 |
| | 10000 | 1.02% | 7.88% | 193703 | 186009 | 63.56 | 50.71 |

Fig. 8 illustrates the unbalanced ratios of partition results of the two algorithms. From the figure, we can see that the unbalanced ratio of small part is better than that of big part for each graph, this is a very natural phenomenon. Most of unbalanced ratios by our algorithm are less than 2%, and most of the results by METIS are between 6% and 9%. Clearly, our algorithm is better than METIS on unbalanced

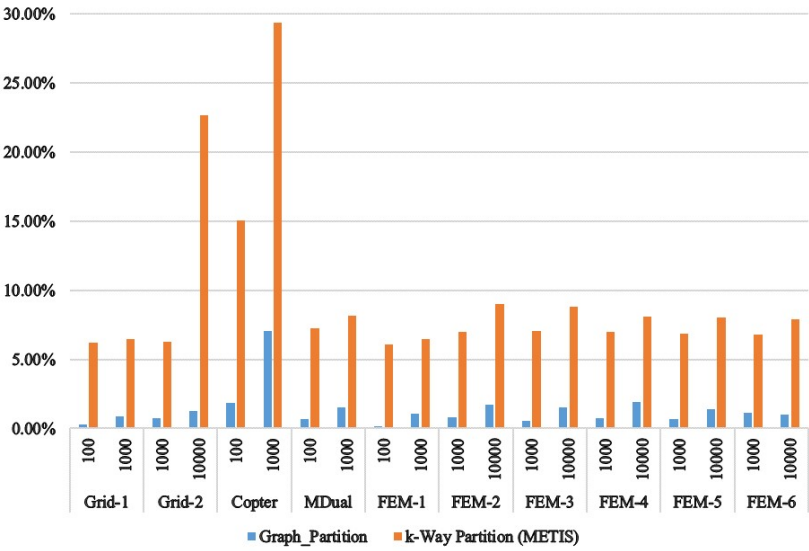


Figure 8 The experimental results on unbalanced ratio.

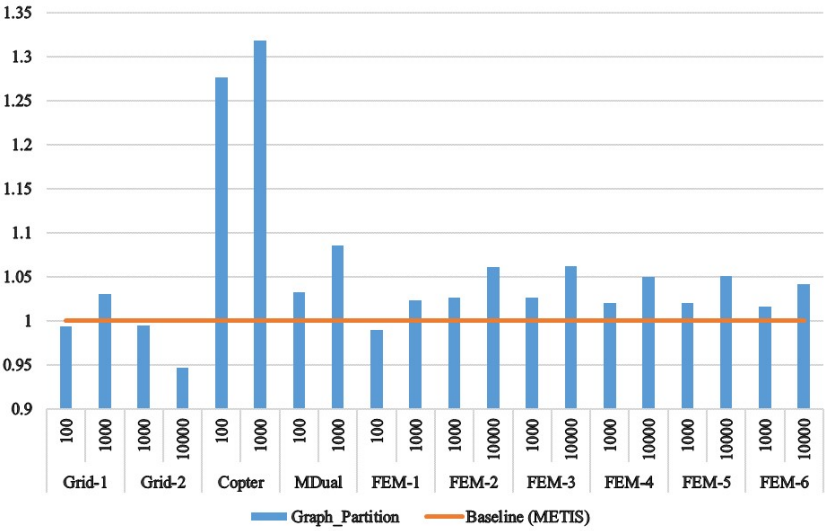


Figure 9 The ratios of max-load of our algorithm to that of METIS.

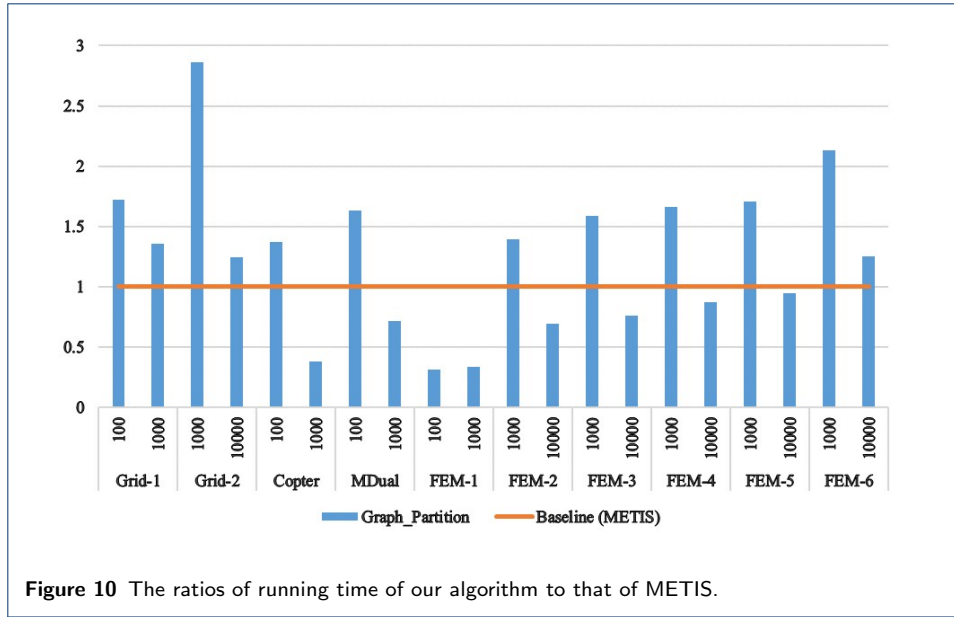


Figure 10 The ratios of running time of our algorithm to that of METIS.

ratio. All unbalanced ratios of graph Copter are worse, the reason is the average degree of Copter is much larger than others.

Fig. 9 and Fig. 10 show the ratios of max-load and running time of our algorithm to that of METIS. Fig. 9 illustrates that most of all ratios of max-load are between 0.94 and 1.06. This implies that there is little difference between the two algorithms in terms of maximum load. Moreover, we can see that the ratio increases with the number of parts, the main reason is that we do not use multi-level modification in back mapping phase. And this is also a key direction in our future work. From Fig. 10, we can see that for the small k , our algorithm often runs longer than METIS; conversely, our algorithm often runs less time than METIS for large k . This difference is related to the number of iterations and the average number of vertices in each part.

5 Conclusions and Future Work

In this paper, we consider the balanced partition problem on large scale directed graphs. Firstly, we present a new mathematical modeling with new objective functions for this problem. Then, we combine multi-level strategy and recursive partition method to design an algorithm to solve it. Finally, by a large number of experiments, we determine the parameters, verify the stability of the algorithm, and compare with k -way partition in METIS in unbalanced ratio, maximum load and running time three aspects. The experimental results show that comparing with METIS, our algorithm is better in unbalanced ratio and has the same quality in maximum load. Furthermore, our algorithm can deal with some graphs with huge scale, which METIS can not return a feasible result.

There are two key points in the future work. The first one is adding modification in back mapping phase, that is, map the partition of D_m back to that of D_0 level by level, and modify the partition of each level to be local optimum. the second point is to ensure the connectivity of each part. Furthermore, finding a new good and efficient graph contraction method is also a meaningful work.

Abbreviations

VLSI: Very large scale integration; BGP: Balanced graph partitioning; RMWM: Random maximum weight matching; RMRM: Random maximum ratio matching; BFD: Best fit decreasing.

Availability of data and materials

The data used or analysed during the current study are available from the corresponding author on reasonable request.

Competing interests

The authors declare that they have no competing interests.

Funding

This work has been supported by National Numerical Windtunnel Project (No. NNW2019ZT5-B16), and Natural Science Foundation of China (Nos. 11871256, 12071194).

Authors' contributions

The contribution of the authors to this work is equivalent. All authors read and approved the final manuscript.

Acknowledgements

The authors would like to thank China Aerodynamics Research and Development Center for providing the practical graph examples.

Author details

¹Department of Mathematics and Statistics, Lanzhou University, Lanzhou, Gansu, 730000, China. ²State Key Laboratory of Aerodynamics, Mianyang, Sichuan, 621000, China. ³Computational Aerodynamic s Institute, China Aerodynamics Research and Development Center, Mianyang, Sichuan, 621000, China.

References

- Andreev, K., Racke, H.: Balanced graph partitioning. *Theory of Computing Systems* **39**(6), 929–939 (2006)
- Feldmann, A.E.: Fast balanced partitioning is hard even on grids and trees. *Theoretical Computer Science* **485**, 61–68 (2013)
- Chekuri, C., Xu, C.: Minimum cuts and sparsification in hypergraphs. *SIAM Journal on Computing* **47**(6), 2118–2156 (2018)
- Xu, B., Yu, X., Zhang, X., Zhang, Z.-B.: An sdp randomized approximation algorithm for max hypergraph cut with limited unbalance. *Science China Mathematics* **57**(12), 2437–2462 (2014)
- Chen, G., Chen, Y., Chen, Z.-Z., Lin, G., Liu, T., Zhang, A.: Approximation algorithms for the maximally balanced connected graph tripartition problem. *Journal of Combinatorial Optimization*. DOI:10.1007/s10878-020-00544-w
- Wu, D., Zhang, Z., Wu, W.: Approximation algorithm for the balanced 2-connected k-partition problem. *Theoretical Computer Science* **609**, 627–638 (2016)
- Chen, Y., Goebel, R., Lin, G., Liu, L., Su, B., Tong, W., Xu, Y., Zhang, A.: A local search 4/3-approximation algorithm for the minimum 3-path partition problem. In: *Proceedings of International Workshop on Frontiers in Algorithmics*, pp. 14–25 (2019)
- Buluç, A., Meyerhenke, H., Safro, I., Sanders, P., Schulz, C.: Recent advances in graph partitioning. In: *Proceedings of Algorithm Engineering*, pp. 117–158 (2016)
- Kernighan, B.W., Lin, S.: An efficient heuristic procedure for partitioning graphs. *The Bell system technical journal* **49**(2), 291–307 (1970)
- Fiduccia, C.M., Mattheyses, R.M.: A linear-time heuristic for improving network partitions. In: *Proceedings of the 19th Design Automation Conference*, pp. 175–181 (1982)
- Chung, F.R., Graham, F.C.: *Spectral Graph Theory*. American Mathematical Soc., Providence, Rhode Island (1997)
- Von Luxburg, U.: A tutorial on spectral clustering. *Statistics and computing* **17**(4), 395–416 (2007)
- Naumov, M., Moon, T.: Parallel spectral graph partitioning. NVIDIA tech. rep (2016)
- Karypis, G., Kumar, V.: A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing* **20**(1), 359–392 (1998)
- Sanders, P., Schulz, C.: Engineering multilevel graph partitioning algorithms. In: *Proceedings of the 19th European Symposium on Algorithms*, pp. 469–480 (2011)
- Tsourakakis, C., Gkantsidis, C., Radunovic, B., Vojnovic, M.: Fennel: Streaming graph partitioning for massive scale graphs. In: *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, pp. 333–342 (2014)

Figures

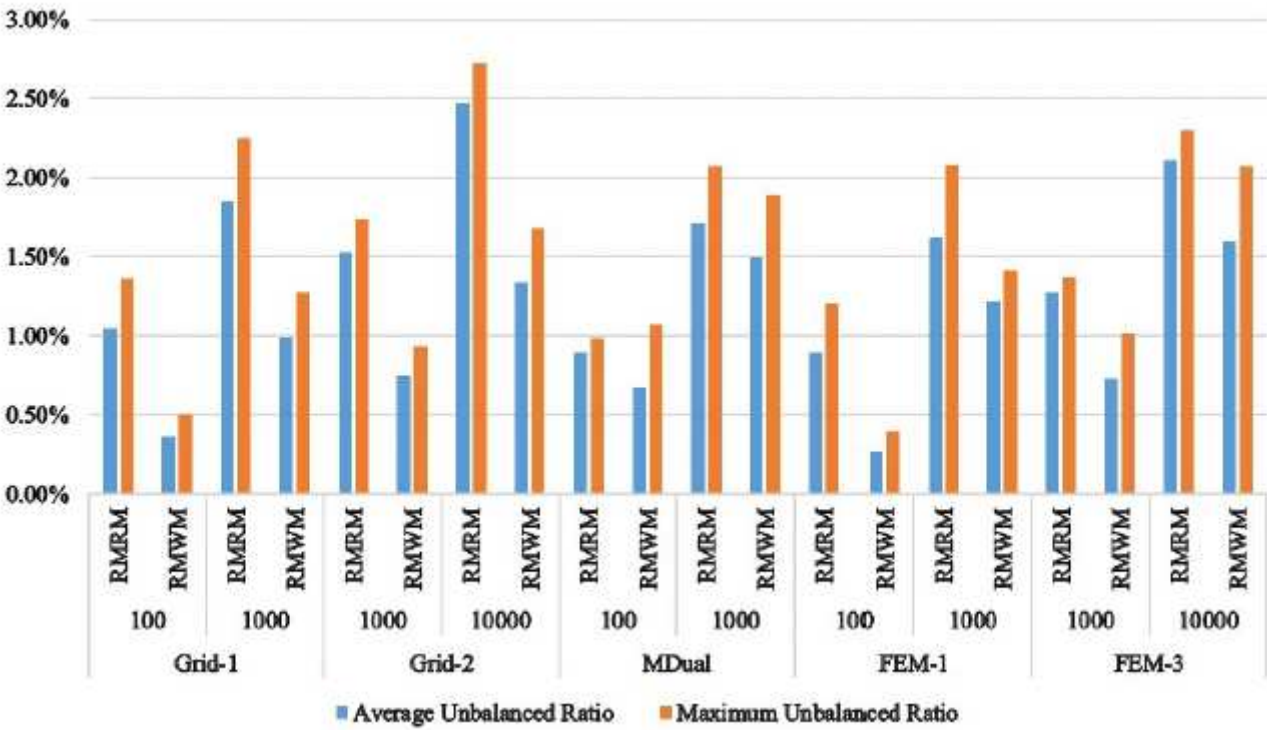


Figure 1

The unbalanced ratios of the two types of maximal matching.

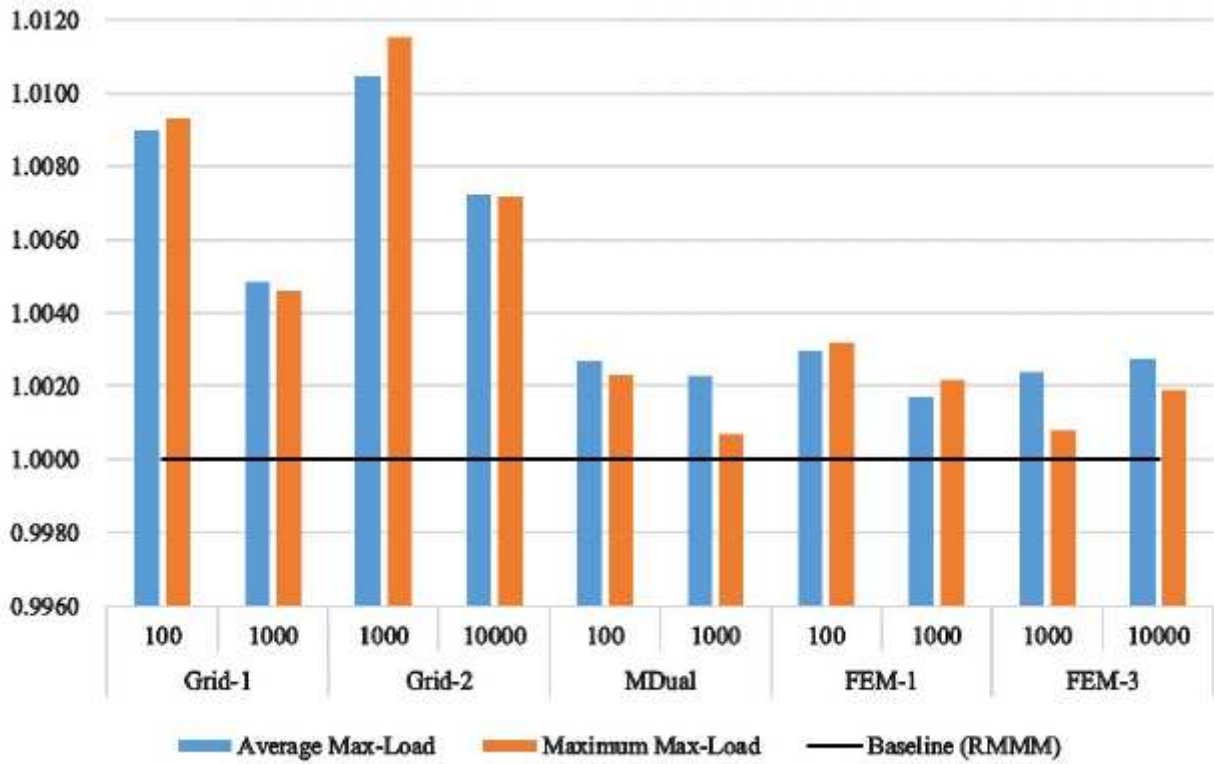


Figure 2

The ratio of the max-load of the RMRM to that of . Bars above the baseline indicate that the performance of RMRM is worse than .

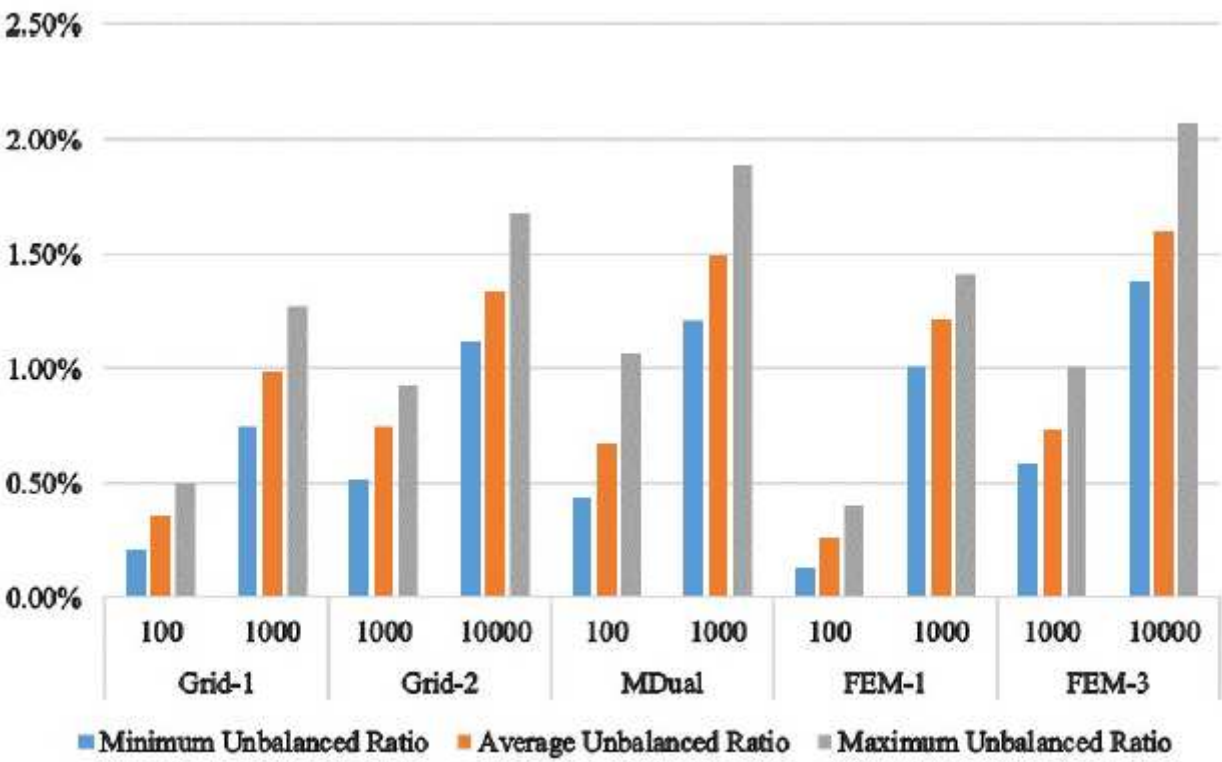


Figure 3

The best, average and worst unbalanced ratios of all examples.

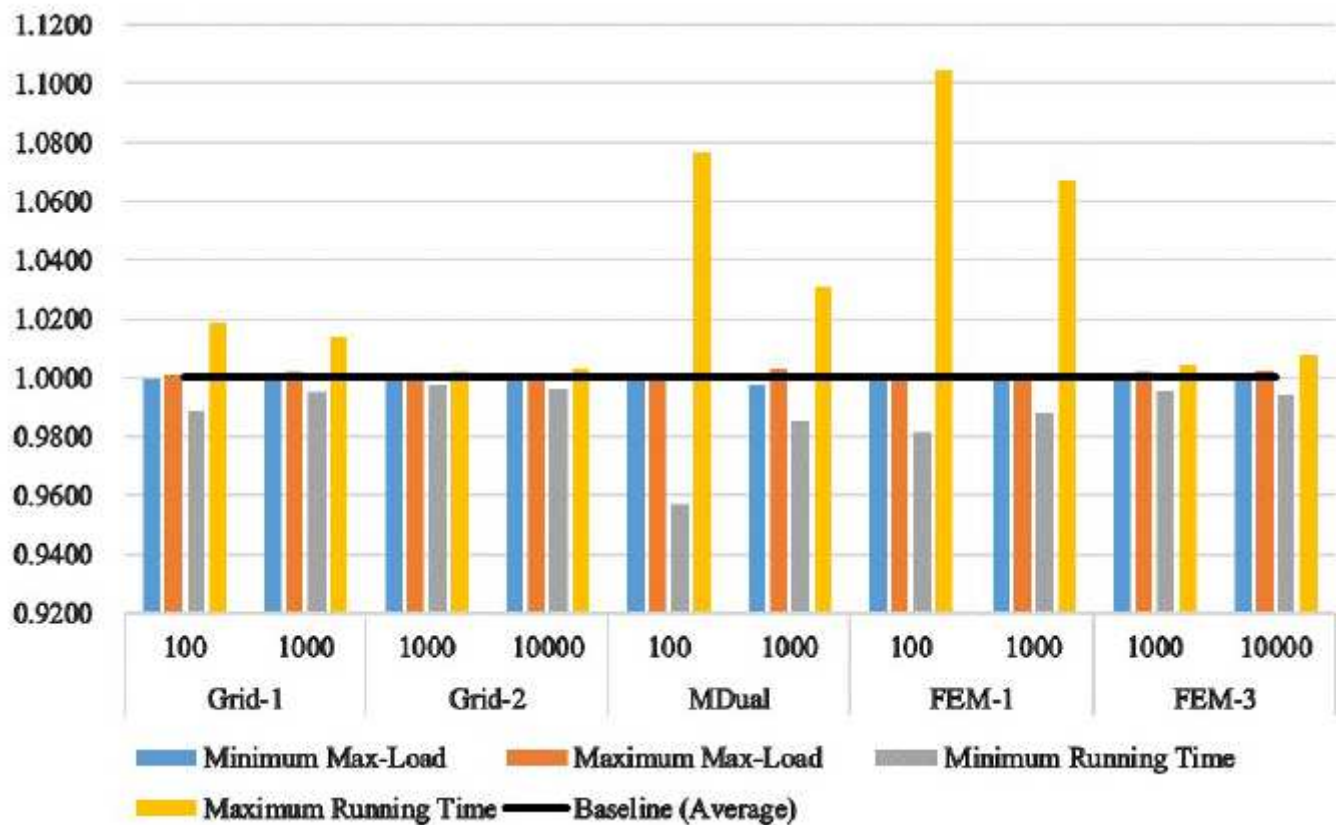


Figure 4

The ratios of best and worst max-load and running time to the relative average results. The baseline is the relative average value.

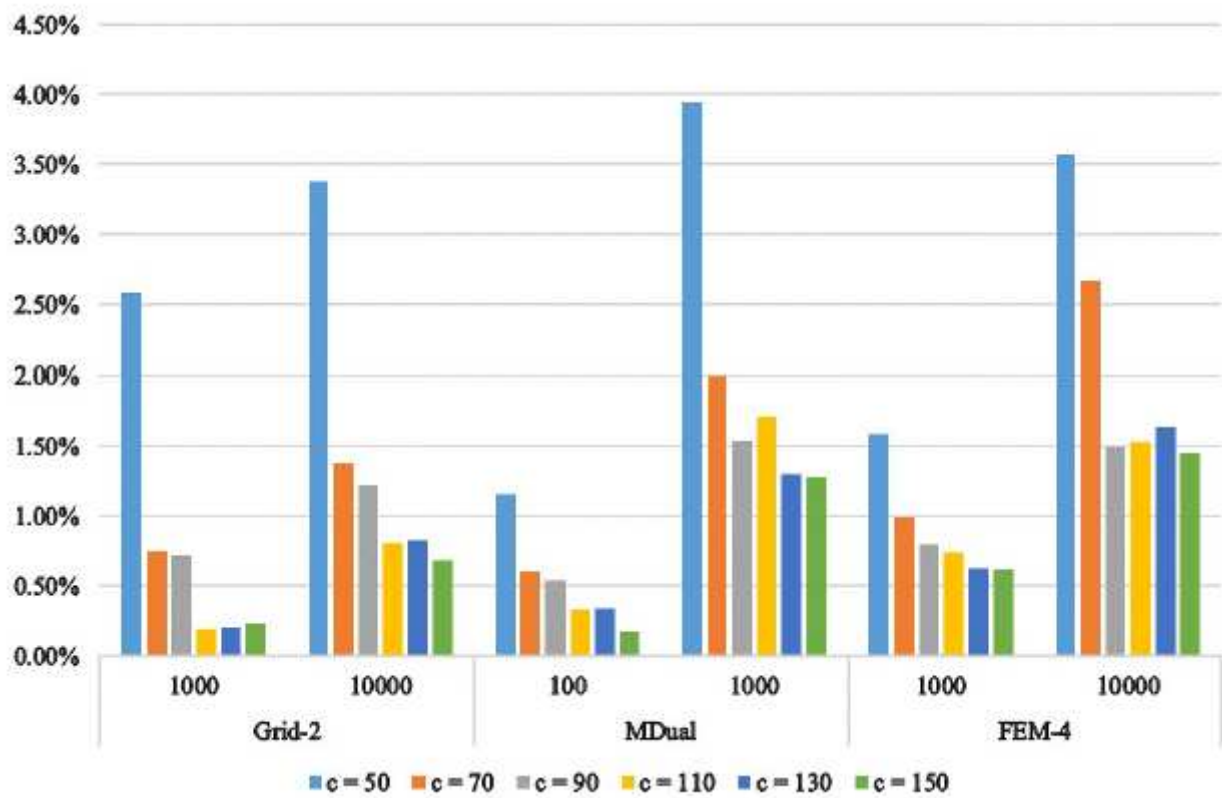


Figure 5

The unbalanced ratios with different contracted parameters.

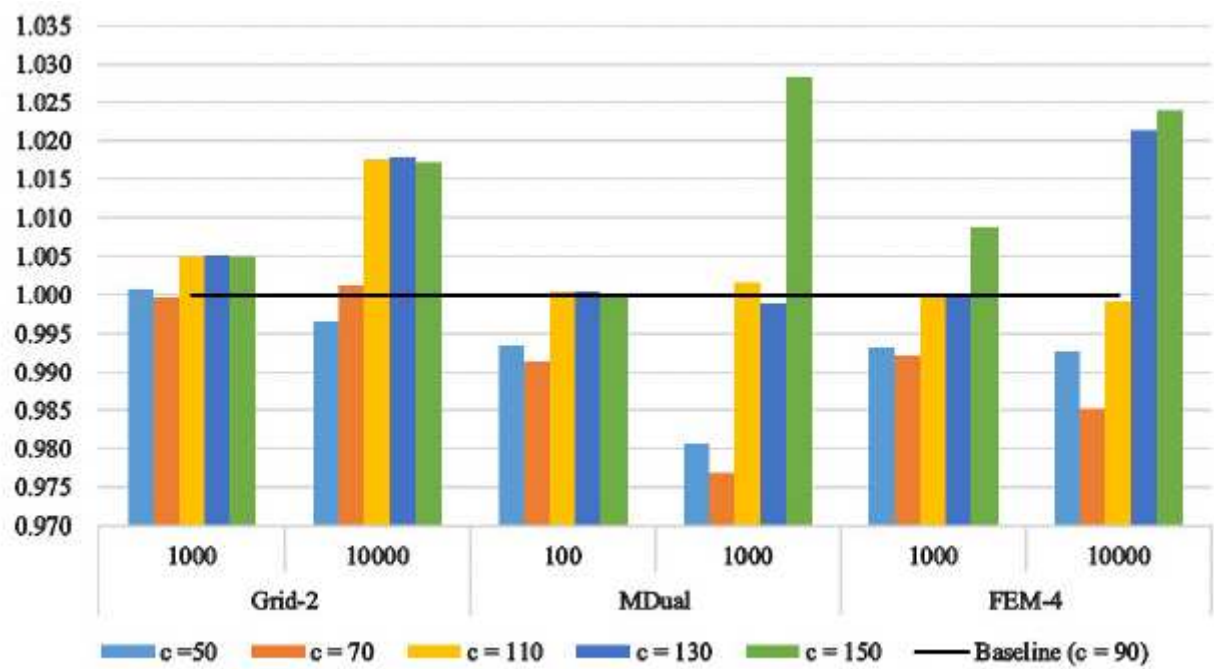


Figure 6

The ratios of max-load of other contracted parameters to that of parameter 90.

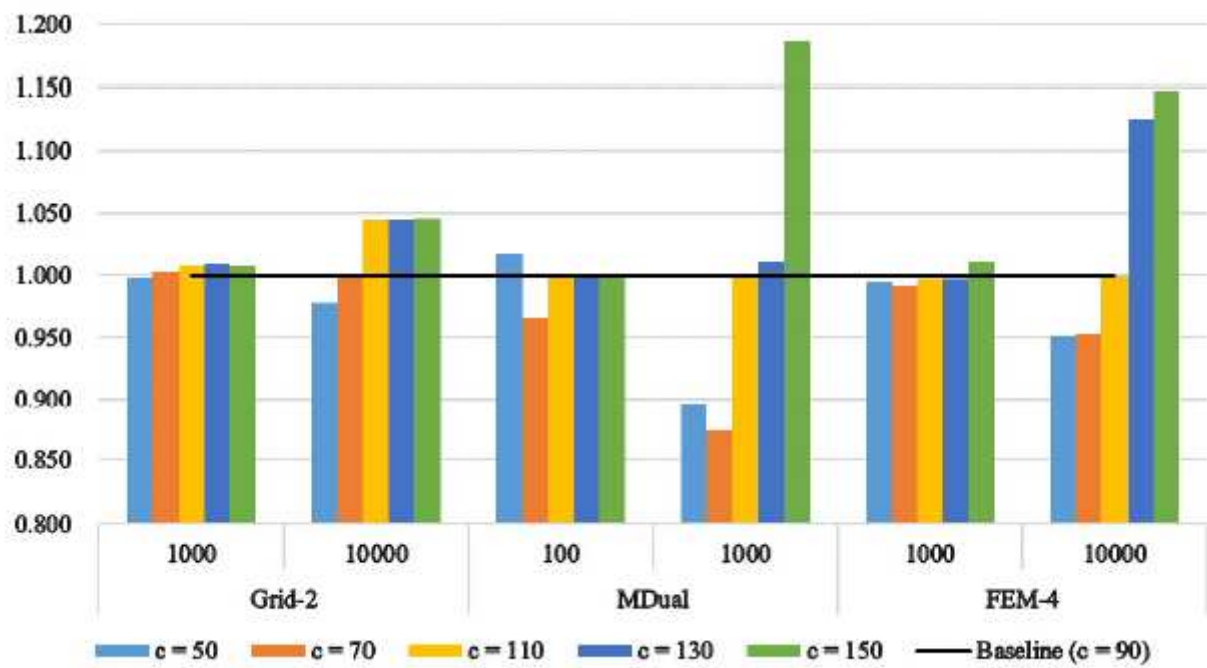


Figure 7

The ratios of running time of other contracted parameters to that of parameter 90.

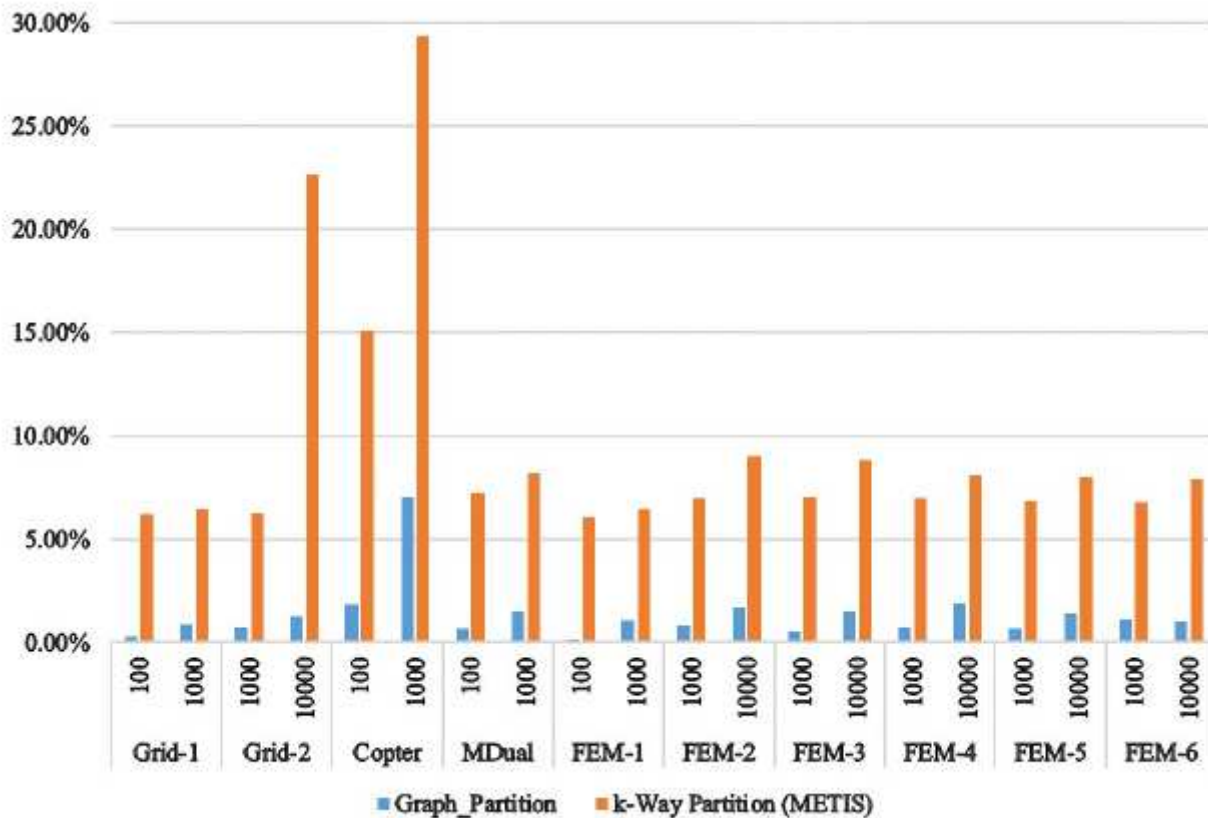


Figure 8

The experimental results on unbalanced ratio.

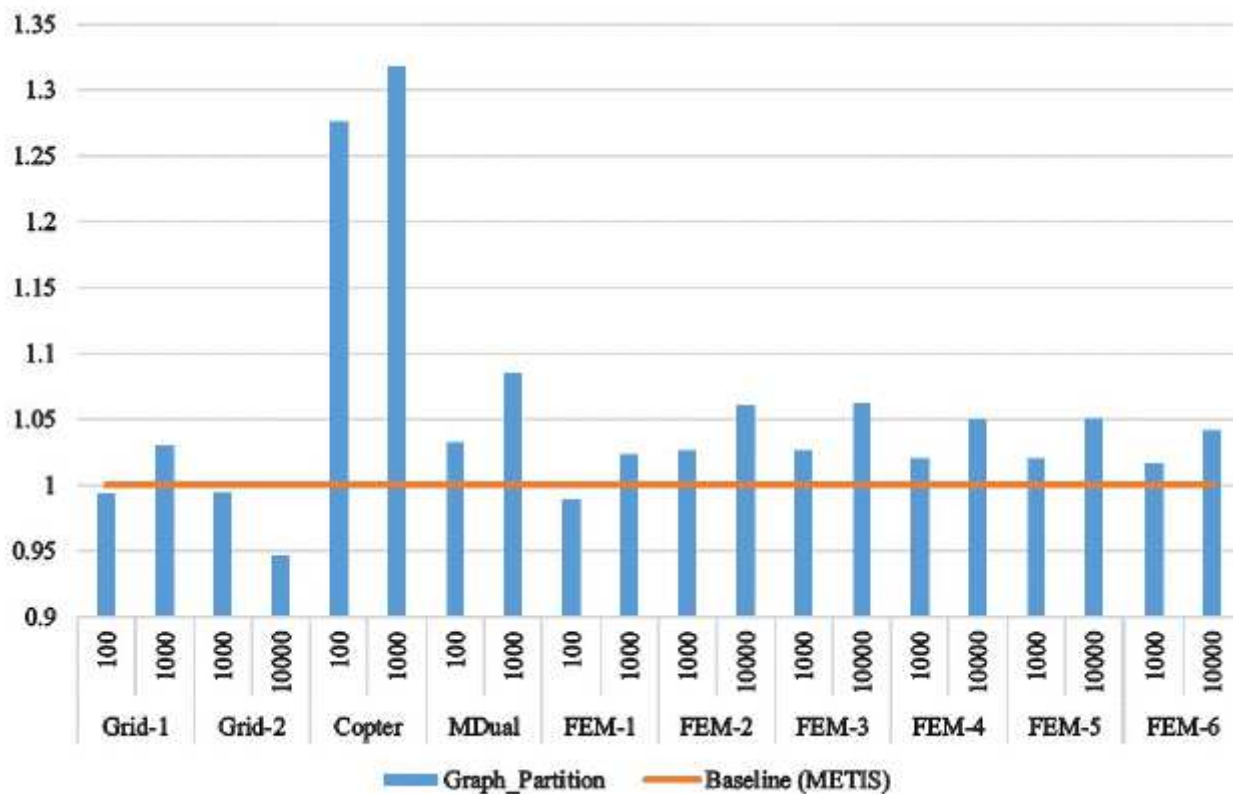


Figure 9

The ratios of max-load of our algorithm to that of METIS.

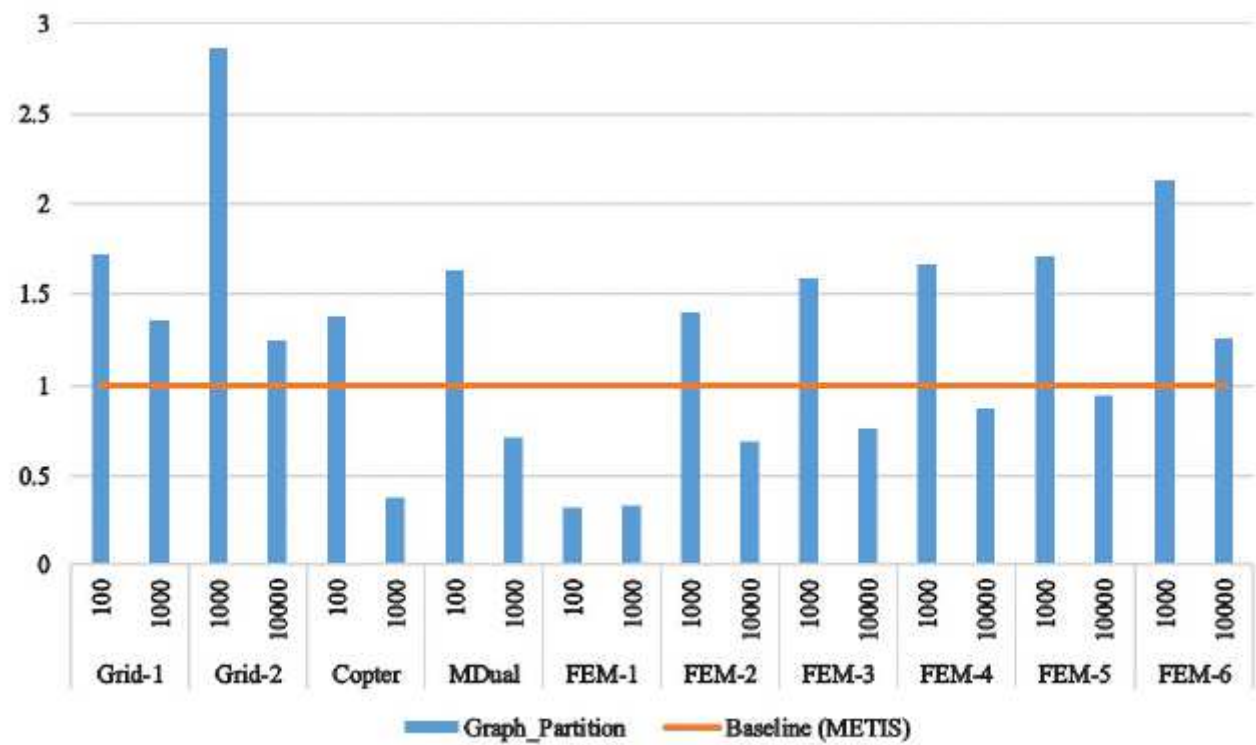


Figure 10

The ratios of running time of our algorithm to that of METIS.