A New Meta-method for Graph Partitioning

Charles-Edmond Bichot

Abstract— **In this paper, a new meta-method based on the physical nuclear process is presented. This meta-method called Fusion-Fission is applied to the two different class of graph partitioning problems. This paper presents results found by this method in comparison with results of classical methods for an air traffic management problem, an image segmentation problem and applied to classical benchmarks. All of these applications of the Fusion-Fission method are successful and the results found by this method outperform state-of-the-art graph partitioning packages both on classical benchmarks and on the air traffic management problem.**

I. Introduction

THIS paper presents a meta-method called Fusion-Fission in reference to the nuclear process on which it is based. This method is particularly adapted to the graph partitioning problem. The Fusion-Fission method is a meta-method in the meaning that it is a method which has been build to explore the space of solution in a hopefully efficient way with the help of an heuristic, *ie* this method can go out of some local minima to continue its search. In regard to these features, this method can also be named a metaheuristic. However, at this time, this method has not been applied to other computational problems than graph partitioning.

The published writing show that there is almost two class of graph partitioning problems. The first one is the "constraint" graph partitioning problem. This class of problem is those of high performance computing [16], [21], [1], [12], [15], [28]. The second class is the "relaxed" graph partitioning problem and is used in image segmentation [22], [20], [18], [23], [10], [27], [17], [2], document clustering [6], [29], [8], [30], air traffic management [3], [4] and many other subjects.

The Fusion-Fission method was initially designed for the "relaxed" graph partitioning class of problems and applied to an air traffic management problem in [3], [4]. This paper presents the last upgrades of this application of Fusion-Fission to solve this problem and introduces two new applications of this method: the first one to image segmentation, the second to high performance computing (*ie.* constraint graph partitioning). In both cases, the application of Fusion-Fission was successful. The quality of the Fusion-Fission algorithm for the relaxed class of problems is illustrated by an image segmentation application and by comparisons with state-of-the-art graph partitioning algorithms on an air traffic management problem. And the quality of the Fusion-Fission algorithm for the constrained class of problems is illustrated by classical benchmarks. Regarding the state-of-the-art graph partitioning algorithms, the results found by the two Fusion-Fission algorithms are very good. All of the results presented in this paper were found using a Debian GNU/Linux Intel Pentium 3GHz workstation.

Charles-Edmond Bichot is with the Laboratoire d'Optimisation Globale, École Nationale de l'Aviation Civile / Direction des Services de la Navigation Aérienne, 7 av. Edouard Belin, Toulouse, France; email: bichot@recherche.enac.fr

The remainder of this paper is organized as follows. Section II describes the Fusion-Fission meta-method, its origin and how it works. Then the Fusion-Fission meta-method is applied to the relaxed graph partitioning problem in section III. In this section, after a short presentation of the relaxed graph partitioning problem (III-A) and the Fusion-Fission application (III-B), the efficiency of the algorithm is illustrated by two examples, the first is an air traffic management problem (III-C) and the second is an image segmentation problem (III-D). In section IV, the Fusion-Fission meta-method is applied to the constrained graph partitioning problem. In this section, the constrained graph partitioning problem is explained (III-A), then the Fusion-Fission application on this problem is presented (IV-B) and finally, some comparisons are made with state-of-the-art partitioning packages on classical benchmarks (IV-C).

II. The Fusion-Fission meta-method

Like many other meta-methods, the Fusion-Fission method comes from real life. This method is based on the nuclear process and particularly the nuclear force and the strong interaction. The nuclear process is a matter reorganization, which can create energy, as in a fission reactor, or destroy energy, as in the fusion process. This energy is based on the nuclear force or residual strong force, which is the force between two or more nucleons (protons or neutrons).

Let us consider a matter state with only nucleons in a plasma of very high temperature which urges nucleons to create nuclei (an atomic nucleus of an atom is a very small dense region of nucleons, thus it is the atom without electrons). Because of the nuclear binding energy, after a long time, all nucleons are binded together to create iron atoms. Nuclear binding energy is the energy required to disassemble a nucleus into separate parts, and it is derived from the strong nuclear force. At the peak of the nuclear binding energy curve, iron is the most tightly-bound nucleus. The iron nucleus is made of 56 nucleons in a range of 2 to 235 for the other nuclei. Because the iron nucleus is very stable, it is more difficult to split it or to join it with another nuclei, than it is with the other nuclei.

The nuclear process may be viewed as a matter reorganization in an optimization process which tends to create atoms with great binding energy. Our idea is to use this process as an optimization process. This meta-method is described as follows:

Let us consider the minimization problem $P = (S, f, \Omega)$, where S is the set of candidates solutions, f is the objective function which assigns to each candidate solution $s \in S$ a cost value $f(s)$, and Ω is a set of constraints. The goal is to find a globally optimal solution $s_{opt} \in S$ which has a minimum cost, and satisfies the constraint Ω. In the graph partitioning problem, S is the set of all possible partitions of a graph $G = (V, E)$ where V is the set of vertices and E is the set of edges, and Ω is the constraint of the size of the parts of the partitions which can be accepted. The classical way to solve this kind of minimization problem is to iteratively search or construct among the set of candidates which satisfy the constraints, $S \cap \Omega$, the candidate which minimizes f. The Fusion-Fission operates differently.

Let us consider the set of constraints, $\Omega = \{\omega_1, \ldots, \omega_n\}$. The idea is to find a neighborhood of Ω for which the computation time and complexity to pass from one neighbor to another is low. A simple way to do that is to construct the neighborhood of Ω with the neighborhood of one constraint ω_i. After the neighborhood of Ω, $\{\Omega_1, \ldots, \Omega_k\}$ is found, the Fusion-Fission process goes iteratively from one minimization problem $(S, f, \Omega_i), i \in \{1, \ldots, k\}$ to another $(S, f, \Omega_j), j \in \{1, \ldots, k\}$. Each minimization problem, P_t (where t is the current step of the iterative algorithm), is solved by a local optimization algorithm, which starts with the solution found by the last minimization problem P_{t-1} adapted to the new minimization problem P_t. The idea is that a good solution in (S, f, Ω_i) is nearby a good solution in (S, f, Ω_j). The choice of the next minimization problem P_{t+1} is random, but must be more often to the original minimization problem (S, f, Ω) than another. The set of candidate solutions S must be as large as possible to accept solutions in the entire Ω's neighborhood.

The similarity with the nuclear process is the following: If we consider a chaotic nuclear organization of the matter, nuclei are split and merged all the time in a way that the matter currently changes its properties. One time, the nuclei are magnesium, the next time hydrogen if they are totally split, or Ununoctium, which is one of the biggest atom known, if nucleons are merged together. At the end of the optimization process, there is only iron nuclei. During the process, if the matter changes its properties, the nature of constraints, or the set of constraints, applied to it changes. This particular vision of a nuclear reorganization which changes the neighborhood of constraints applied to it, conducted to create the Fusion-Fission meta-method.

III. Application to relaxed graph partitioning

The Fusion-Fission method was first designed for the relaxed graph partitioning problem. Its first application is an air traffic management problem. First, this section presents the relaxed graph partitioning problem. Then, the Fusion-Fission method applied to the relaxed graph partitioning problem is introduced. Then, two problems are presented to illustrate the Fusion-Fission efficiency, the first is an air traffic management problem and the second is an image segmentation problem.

A. The relaxed graph partitioning problem

The general problem of graph partitioning is to partition the set of vertices V of a graph $G = (V, E)$, where E is the set of edges, into several vertex subsets or parts, respecting constraints, while minimizing an objective function. Depending on goals, constraints are different.

The relaxed graph partitioning problem consists in partitioning a graph into parts of different sizes which must be in a range of values (the imbalance constraint), while minimizing the normalized cut ($Ncut$) objective function.

Let us describe more formally the relaxed graph partitioning problem. Let $G = (V, E)$ be an undirected weighted graph with a set of vertices $V = \{V_i, i = 1 \ldots n\}$ and a set of edges E. For all vertices v_i in V, let $w(v_i)$ be its weight. For all couple of vertices (i, j) in E, let $w(i, j)$ be its weight.

Given a partitioning of the vertex set V into two subsets V_1 and V_2, the cut between them is :

$$cut\,(V_1, V_2) = \sum_{i \in V_1, j \in V_2} w(i, j) \tag{1}$$

The normalized cut objective function introduced in [23] is defined as follows :

$$Ncut\,(\pi_k) = \sum_i \frac{cut\,(V_i, V - V_i)}{cut\,(V_i, V)} \tag{2}$$

Minimizing the normalized cut objective function is equivalent for each part to minimize the edge cut between this part and the others, and to maximize the sum of the edges weights between vertices of this part.

The imbalance of a partition π_k is the ratio between the maximum size of a part and the average size of a part. The $Imbalance$ constraint is defined as follows :

$$\max_{V_i \in \pi_k} \left(\sum_{v \in V_i} w(v) \right) \leq Imbalance * \left\lceil \sum_{v \in V} \frac{w(v)}{k} \right\rceil \tag{3}$$

where the function $f : x \to \lceil x \rceil$ returns the smallest integer greater than x.

In relaxed graph partitioning problems, $Imbalance$ is usually between 1.5 and 3.0. This means that, in a partition, parts can be of very different sizes.

B. Fusion-Fission application

This paper shortly presents the application of the Fusion-Fission meta-method to the relaxed graph partitioning problem. For a detailed presentation, see [4] and [5].

The junction between the Fusion-Fission meta-method and the relaxed graph partitioning problem is easy to understand if we consider the nuclear process which is at the origin of the meta-method. The nuclear process involves nuclei and nucleons. Nuclei are split and merged by the nuclear process, and nucleons are going from nuclei to nuclei. A simple junction between the nuclear process and the graph partitioning problem is to compare nuclei with subsets of vertices (or parts) and nucleons with vertices.

The physical nuclear optimization process tends to construct iron nuclei by fusion and fission of nuclei. Similarly,

the Fusion-Fission algorithm iteratively creates partitions of different number of parts, by fusion and fission of parts. This partitions are converging towards a partition which minimizes the normalized cut objective function under the imbalance constraint.

The Fusion-Fission method consists in repeating a perturbation process to a partition of the graph till a stop condition is reached. The perturbation process starts with a partition of the graph and successively applies fusion or fission to the partition.

A nuclear reaction can only be created in a very high temperature plasma. Thus the Fusion-Fission method includes a "temperature" which is used to control the process. The temperature is used to stop the perturbation process after a chosen number of steps.

In the nature, when a fusion or a fission is done, some nucleons are ejected by the process. These nucleons join another nuclei, or make other fissions if they have a very high energy. The number of ejected nucleons is chosen randomly according to "rules". Then some "rules" are defined to eject randomly in a fusion or in a fission some vertices of its part. These rules are automatically adjusted during the execution of the algorithm by a self-learning function. A rule is just a probability function to eject a certain amount of vertices of the part. Vertices which lowly connected with vertices of the same part are ejected first.

The process of fusion consists in merging two parts. The new part is the contraction of the two former parts minus the random number of vertices "ejected" by the rule. Then, these vertices are aggregated to the part with which they are the most connected.

The process of fission is much more complex. First, like in the fusion process, some vertices can be "ejected" of the part. After that, the part is split into two parts with the help of a graph bisection algorithm. The graph bisection algorithm which has been chosen is an agglomerative method based on the percolation process [4]. When the part is split, the vertices ejected are aggregated to a partition with which they are highly connected, or they can split other parts in a chained fission reaction.

Comparing with other graph partitioning methods, one particularity of the Fusion-Fission method is that it finds partitions with different number of parts in one run. Thus, in a Fusion-Fission process partitions number of parts are confined to a range centered on an average number of parts, k.

C. Air Traffic Management partitioning

The first application of the Fusion-Fission method to the relaxed graph partitioning problem was an air traffic management problem. This problem is a reorganization of the European airspace partition.

Each air traffic controller supervises a limited space, called an air traffic sector. Controllers have qualifications to work only on a set of sectors. These sets are called functional airspace blocks. The problem consists in cutting the European airspace into blocks.

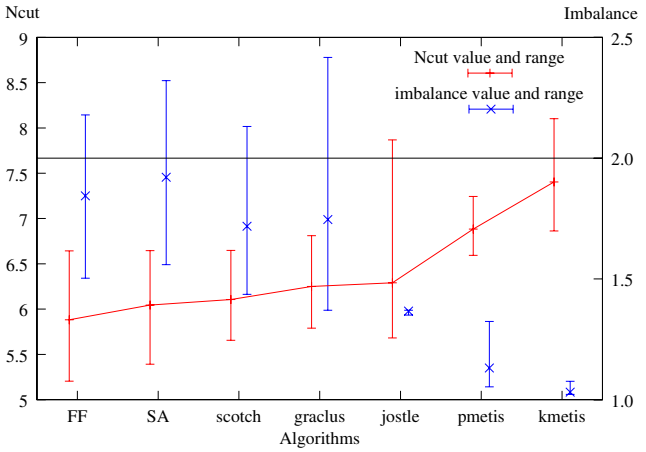

Fig. 1. Algorithms $Ncut$'s average and range results of 100 permutations of the air traffic management graph

Because "it is well known that controller-controller coordination is easier and more effective inside an air traffic control unit (a block) than between air traffic control units" [11], we search to maximize flows of aircraft inside blocks and minimize flows of aircraft between blocks. The graph vertices are air traffic sectors and edges are flows of aircraft between sectors. The air traffic problem is a relaxed multi-way graph partitioning problem which minimizes the normalized cut objective function $Ncut$. The number of parts of the partition is the number of functional airspace blocks into which we want to cut the European airspace.

The Fusion-Fission algorithm presented in subsection III-B is compared with other state-of-the-art graph partitioning packages: Scotch [19], Graclus [7], Jostle [26], Metis [14], [15], and with the simulated annealing meta-method applied to graph partitioning as it was explained in [13].

A graph has been made from a day of traffic in Europe. This graph is composed of 762 vertices and $3,165$ edges. The graph is stored using the Chaco input graph format also used by the other algorithms. Because we do not have data of hundred of days of traffic, one hundred new files are computed from the original graph. To create these files, the order in which the graph is described in the original graph file (the indices of vertices) is randomly permuted.

The algorithms are compared on figure 1. The left axis shows average and range $Ncut$ results of one hundred permutations of the air traffic management graph file. The algorithms are classified from the best average $Ncut$ results to the worst. The Fusion-Fission algorithm performs better than the others, followed by the simulated annealing algorithm. Because multilevel algorithms are always better than other methods in graph partitioning, it can be surprising that they return worse results than the meta-methods. Two reasons can explain it. The first is that only Scotch and Graclus packages are designed for the relaxed graph partitioning problem, the second is that the multilevel methods compute in less than a second when the meta-method are limited to two minutes.

The right axis of figure 1 shows average and range

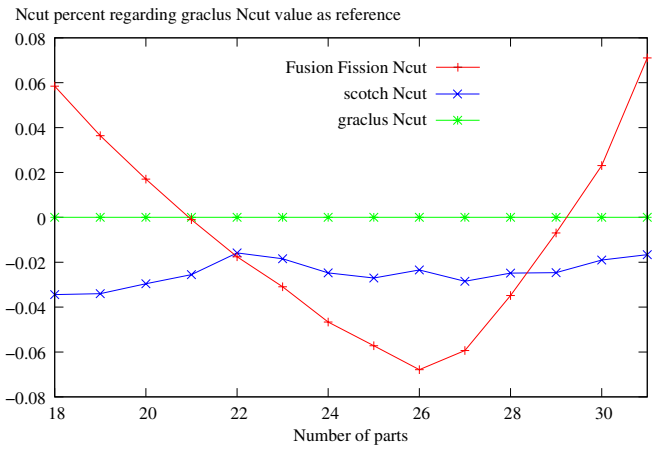

Fig. 2. Comparisons between Fusion-Fission results for $k = 26$, with Graclus and Scotch results for $k \in \{18, \ldots, 31\}$. Each algorithm results are presented as rate variation between the algorithm's $Ncut$ average and the corresponding $Ncut$ average of Graclus

$Imbalance$ results. The line $y = 2$ is the threshold above while the imbalances of the partitions are considered too high regarding air traffic constraints. The simulated annealing algorithm returns partitions with very high imbalances, and the Jostle package returns partitions of very different imbalances. Imbalances of the partitions returned by Jostle and Metis packages show us that these packages are not designed for the relaxed graph partitioning problem. To conclude, the Fusion-Fission algorithm returns the best partitioning results with an imbalance which is tolerable, and the Scotch package is the best multilevel algorithm regarding partitions results.

Figure 2 presents a comparison between one run of the Fusion-Fission algorithm for $k = 26$ with Graclus and Scotch results for $k \in \{18, \ldots, 31\}$. Each algorithm results are presented as rate variation between the algorithm's $Ncut$ average and the corresponding $Ncut$ average of Graclus. The partitions found by the Fusion-Fission algorithm are better than those of Scotch for $k \in \{22, \ldots, 28\}$. And the partitions found by the Fusion-Fission algorithm outperforms the partitions found by Graclus for $k \in \{21, \ldots, 29\}$.

D. Image segmentation

Following the work of Jianbo Shi and Jitendra Malik [23], the Fusion-Fission algorithm is applied to the perceptual grouping problem in vision. Therefore, the image segmentation problem is treated as a relaxed graph partitioning problem which minimizes the $Ncut$ objective function.

One of the approach of image segmentation consists in converting an image into a graph. This conversion must follow two different steps.

In the first step, an image is converted into a matrix of intensity. To simplify, only grayscale images are used in this paper. Grayscale images intended for visual display are typically stored with 8 bits per sampled pixel, which allows 256 intensities. Thus pixels are only represented as numbers between 0 and 255 in the intensity matrix.

Fig. 3. Comparisons between the original image (on the left) and the segmented image (on the right)

During the second step, a graph is constructed with a number of vertices equal to the number of pixels of the image. The edges of this graph are created as follows. An edge between two vertices exists only if the distance between the two corresponding vertices in the image is not to long. The weight of the edges are their intensity similarity balanced with a distance function:

$$\forall (i,j) \in E, w(i,j) = \exp^{\frac{-\| X(i)-X(j) \|_2^2}{\sigma_X}} * \exp^{\frac{-\| I(i)-I(j) \|_2^2}{\sigma_I}}$$

where $\| X(i) - X(j) \|_2$ is the Euclidean distance between the pixels corresponding to i and j, and $\| I(i) - I(j) \|_2$ is metric between intensity of pixel i and pixel j, normalized between 0 and 1. σ_X and σ_I are weighting parameters.

The number of parts in which an image should be segmented is subjective. This is illustrated by the Berkeley Segmentation Dataset[1]. In this database, some people have segmented image with no constraint on the number of parts. For each image, the results show a great diversity of the number of parts found. In this case, the property of the Fusion-Fission algorithm which is able to partition a graph into different numbers of parts in one run helps to find the most appropriate segmentation of the image. This is an advantage compared to other algorithms, as the spectral algorithm used by Jianbo Shi and Jitendra Malik.

Figure 3 presents results between the original image and the segmented image of three images of the Berkeley Segmentation Dataset. The ostrich image is difficult to segment and the algorithm returns bad partitions for a number of parts lower than four. However, the Fusion-Fission algorithm returns good segmentations for the two other images. The computation time to find these results is less than a minute.

[1] www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds

IV. Application to constrained graph partitioning

Graph partitioning methods are mainly designed to solve the "constrained" graph partitioning problem. Then, to show the efficiency of the Fusion-Fission meta-method it is important to compare its application on this problem with state-of-the-art algorithms.

This section first presents the constrained graph partitioning problem. Then the Fusion-Fission method is applied to the constrained graph partitioning problem. After that, tests and comparisons on classical benchmarks are presented.

A. The constrained graph partitioning problem

As it is explained in subsection III-A, the general graph partitioning problem is to partition a graph into several parts, respecting constraints, while minimizing an objective function. This subsection uses the notations of subsection III-A.

The classical graph partitioning problem consists in fixing the number of parts, finding parts with roughly the same weights, and minimizing the sum of the weight of the edges connecting vertices of different parts. In this paper, we name this problem the "constrained" graph partitioning problem as opposed to the "relaxed" graph partitioning problem. In the literature, this problem is named the multi-way or k-way graph partitioning problem. It is widely used in VLSI design and parallel computing.

Formally, the k-way graph partitioning problem consists in finding a partition $\pi_k = \{V_1 \dots V_k\}$ of a graph $G = (V, E)$ into k parts which respects a hard imbalance constraint and minimizes the Cut objective function:

$$Cut\,(\pi_k) = \sum_{i<j} cut\,(V_i, V_j) \tag{4}$$

where $cut\,(V_i, V_j)$ is defined in equation 1.

The imbalance value, $Imbalance$, of a partition is defined in equation 3. As opposed to the relaxed graph partitioning problem, in the constrained problem the imbalance of a partition is hard, *i.e.* $Imbalance \leq 1.05$. It is known that a classical way to find a better partition regarding the objective function, is to increase the imbalance [24]. If $Imbalance = 1$ then the graph is perfectly balanced, *i.e.* every part has the same weight. Else, if $Imbalance > 1$, the Cut objective function's value tends to decrease rapidly. Classical benchmarks are tested with $Imbalance \in \{1.0; 1.01; 1.03; 1.05\}$.

Formally, the set of constraints Ω of the k-way graph partitioning problem is:

- The number of parts of the partition must be k.
- $\bigcup_i V_i = V$.
- $\forall i \neq j, V_i \cap V_j = \emptyset$.
- Each part V_i must respect the imbalance of the equation (3).

B. Fusion-Fission application

Section II presents the Fusion-Fission meta-method as a minimization problem subject to constraints. With the notation of the section II, a neighborhood of the set of constraints Ω has to be found. A neighborhood of Ω can easily be constructed by creating a neighborhood of the constraint on the number of parts of the partition, k. This neighborhood consists in finding partitions with $k \in \{k - l, \dots, k + l\}$, where l is an integer. This constraint is chosen because in some problems, k is not fixed. Then the Fusion-Fission method is able to find partitions for a wide range of parts number. This is not possible with classical graph partitioning methods for which each partition must be computed separately.

At each iteration of the Fusion-Fission process, a new partition π_p^t of the graph based on the last partition π_q^{t-1} has to be created. The old partition π_q^{t-1} has q parts and the new partition π_p^t must have p parts. To create a partition of p parts, starting with a partition of q parts, the algorithm starts by splitting each of the q parts of π_q^{t-1} into p parts. The pMetis package [14] is used to make these new partitions. The pMetis package is the most popular graph partitioning package, and we use it because of its efficiency and its rapidity. Then, with the $p*q$ parts created, a new partition of p parts is done by collapsing the parts together. The $p*q$ parts are viewed as $p*q$ vertices of a new graph. Then the pMetis package is used to partition this graph into p parts. After a refinement step which uses a Kernighan-Lin algorithm [16], [9], the new partition π_p^t is computed.

Graph name	$Card(V)$	$Card(E)$	description
add20	2395	7462	20-bit adder
data	2851	15093	
3elt	4720	13722	2D nodal graph
uk	4824	6837	2D dual graph
add32	4960	9462	32-bit adder
bcsstk33	8738	291583	3D stiffness matrix
whitaker3	9800	28989	2D nodal graph
crack	10240	30380	2D nodal graph
wing_nodal	10937	75488	3D nodal graph
fe_4elt2	11143	32818	
vibrobox	12328	165250	Sparse matrix
bcsstk29	13992	302748	3D stiffness matrix
4elt	15606	45878	2D nodal graph
fe_sphere	16386	49152	
cti	16840	48232	3D semi-struct. matrix
memplus	17758	54196	Memory circuit
cs4	22499	43858	3D dual graph
bcsstk30	28924	1007284	3D stiffness matrix
bcsstk31	35588	572914	3D stiffness matrix
bcsstk32	44609	985046	3D stiffness matrix
t60k	60005	89440	2D dual graph
wing	62032	121544	3D dual graph
brack2	62631	366559	3D nodal graph

TABLE I
DESCRIPTION OF THE 23 TEST GRAPHS

C. Tests on classical Benchmarks

The test graphs have been chosen to be a representative sample of medium scale real-life problems. All of these graphs can be downloaded from the graph partitioning archive of Chris Walshaw [25] at http://staffweb.cms.gre.ac.uk/∼c.walshaw/partition. Table I gives a list of the 23 graphs with their number of vertices, their number of edges, and a short description. Like most

k	Software	FF better	FF equal	FF lower
2	Jostle	22	1	0
	pMetis	22	1	0
4	Jostle	23	0	0
	pMetis	21	0	2
8	Jostle	20	1	2
	pMetis	18	4	1

TABLE II

PRESENTS THE NUMBER OF TIMES THE FUSION-FISSION ALGORITHM RETURNS PARTITIONS WHICH ARE IN AVERAGE OF THE 20 TESTS, BETTER, EQUAL OR LOWER IN QUALITY THAN THE JOSTLE OR THE PMETIS ALGORITHMS

of partitions graph benchmarks, all of this graphs are vertex and edge unweighted.

The Fusion-Fission algorithm presented in subsection IV-B is restricted to $1,000$ iteration of the main process. Thus the maximum computation time is obtained for the partitioning of bcsstk30 into 8 parts with 2 and half hours of computation. The minimum computation time is obtained for the partitioning of add20 into 2 parts with one and half minute of computation.

To assess the quality of the Fusion-Fission algorithm, its results are compared with the results returned by two most popular graph partitioning packages: Jostle [26] and pMetis [14]. Each graph is stored using the Chaco input graph format also used by pMetis and Jostle. To present relevant results, the robustness of each algorithm is tested. For each graph, the algorithms runs 20 times the same graph presented differently. Indeed, for each graph file, the order in which the graph is described in the file is randomly changed to create 20 new graph files.

The average and range of the Cut's partitions results are presented in figures 4 to 6. Figures 4, 5 and 6 show the partitions Cut results into respectively 2, 4 and 8 parts returned by the Fusion-Fission, the Jostle and the pMetis algorithms. The results are presented as rate variation with the pMetis average Cut's partition results. In the figures, the lines show average Cut's partition results of each algorithm, and the vertical bars show the Cut's partition ranges returned by each algorithm.

Results presented in figures 4 to 6 show that the Fusion-Fission algorithm performs better than the two state-of-the-art partitioning packages. Table II presents a comparison between the number of times the Fusion-Fission algorithm returns partitions which are, in average of the 20 tests, better, equal or lower in quality than the Jostle or the pMetis algorithms.

Note that differences in quality tend to decrease as the number of parts increases. This is partially due to two facts. The first is, when the number of parts is decreasing, the number of vertices per part is increasing, and then, the margins for difference between the partitions cut is increasing. Indeed, there are fewer configurations of partitions for a great number of parts, than for a small number of parts. The second is due to the weakness of our algorithm for a greater number of parts.

It can be noticed that the Fusion-Fission algorithm is the most robust algorithm. Indeed, in most of the cases, its Cut's value range is lower than the lower bound of the Cut's value range of the other algorithms. As it was noticed in subsection III-C, Jostle is the less robust algorithm.

V. CONCLUSION

This paper presents a new meta-method called Fusion-Fission. This method is based on the physical nuclear process. This new meta-method is applied to the relaxed and the constrained graph partitioning class of problems. The relaxed graph partitioning class of problems is illustrated by an air traffic management problem and an image segmentation problem. The constrained graph partitioning class of problems is illustrated by some tests on classical benchmarks. In the two cases, Fusion-Fission results are presented in comparisons with results returned by state-of-the-art graph partitioning packages. On these applications, the two Fusion-Fission algorithms outperforms the other algorithms for the quality of their results, however their computation times are quite longer. Since the Fusion-Fission meta-method has been successfully applied to the graph partitioning problem, other application areas should be investigated in future works to extend the application range of this method, and maybe to propose a new metaheuristic.

REFERENCES

[1] Stephen T. Barnard and Horst D. Simon. A fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. *Concurrency: Practice and Experience*, 6:101–107, 1994.

[2] Florence Benezit, Timothee Cour, and Jianbo Shi. Spectral segmentation with multi-scale graph decomposition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[3] Charles-Edmond Bichot. A metaheuristic based on fusion and fission for partitioning problems. In *Proceedings of the 20th IEEE International Parallel and Distributed Processing Symposium*, 2006.

[4] Charles-Edmond Bichot. A new method, the fusion fission, for the relaxed k-way graph partitioning problem, and comparisons with some multilevel algorithms. *Journal of Mathematical Modeling and Algorithms (JMMA)*, 6(3):319–344, 2007.

[5] Charles-Edmond Bichot, Jean-Marc Alliot, Nicolas Durand, and Pascal Brisset. Optimisation par fusion et fission. application au problme du dcoupage arien europen. *Journal Europen des Systmes Automatiss (JESA)*, 38(9-10):1141–1173, 2004.

[6] Inderjit S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 269–274, 2001.

[7] Inderjit S. Dhillon, Yuqiang Guan, and Brian Kullis. Kernel k-means, spectral clustering, and normalized cuts. In *Proceedings of the 10th ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 551–556, 2004.

[8] Chris Ding, Xiaofeng He, Hongyuan Zha, Ming Gu, and Horst D. Simon. A min-max cut algorithm for graph partitioning and data clustering. In *Proceedings of the IEEE International Conference on Data Mining*, pages 107–114, 2001.

[9] C. M. Fiduccia and R. M. Mattheyses. A linear-time heuristic for improving network partitions. In *Proceedings of 19th ACM/IEEE Design Automation Conference*, pages 175–181, 1982.

[10] Yoram Gdalyahu, Daphna Weinshall, and Michael Werman. Self-organization in vision: Stochastic clustering for image segmentation, perceptual grouping, and image database organization. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 23(10):1053–1074, 2001.

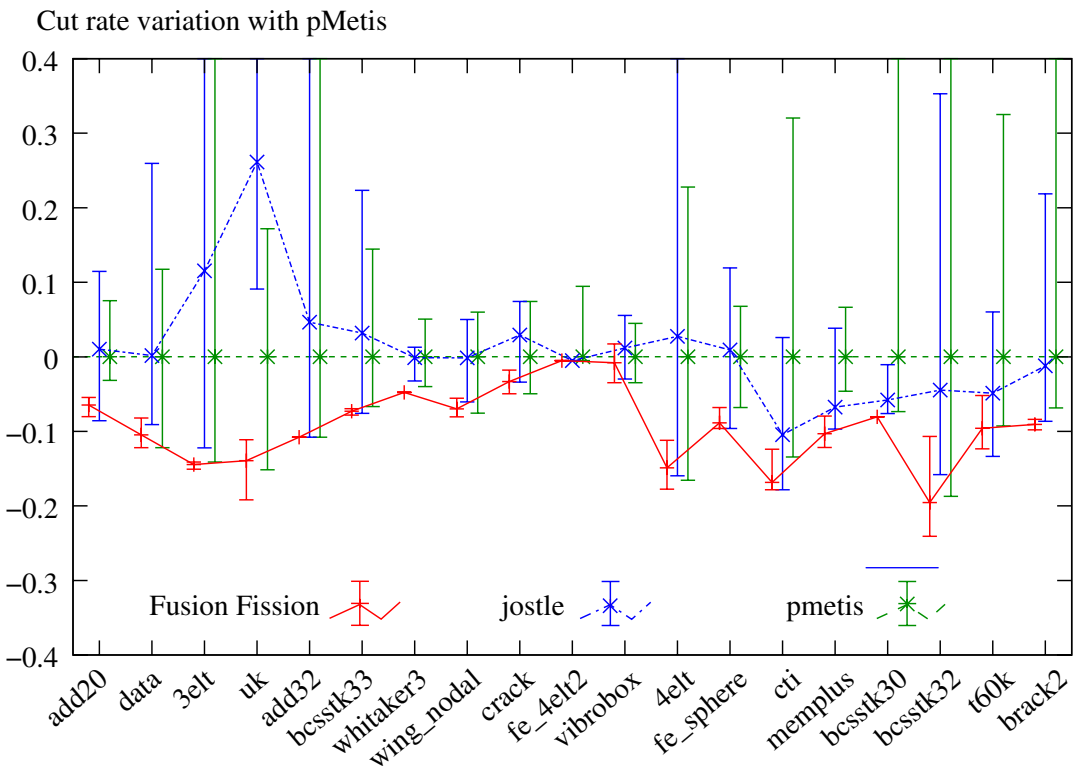

Fig. 4. Average and range Cut's rate variation results of 2 parts partitions returned by Fusion-Fission, Jostle and pMetis, with corresponding pMetis average Cut.

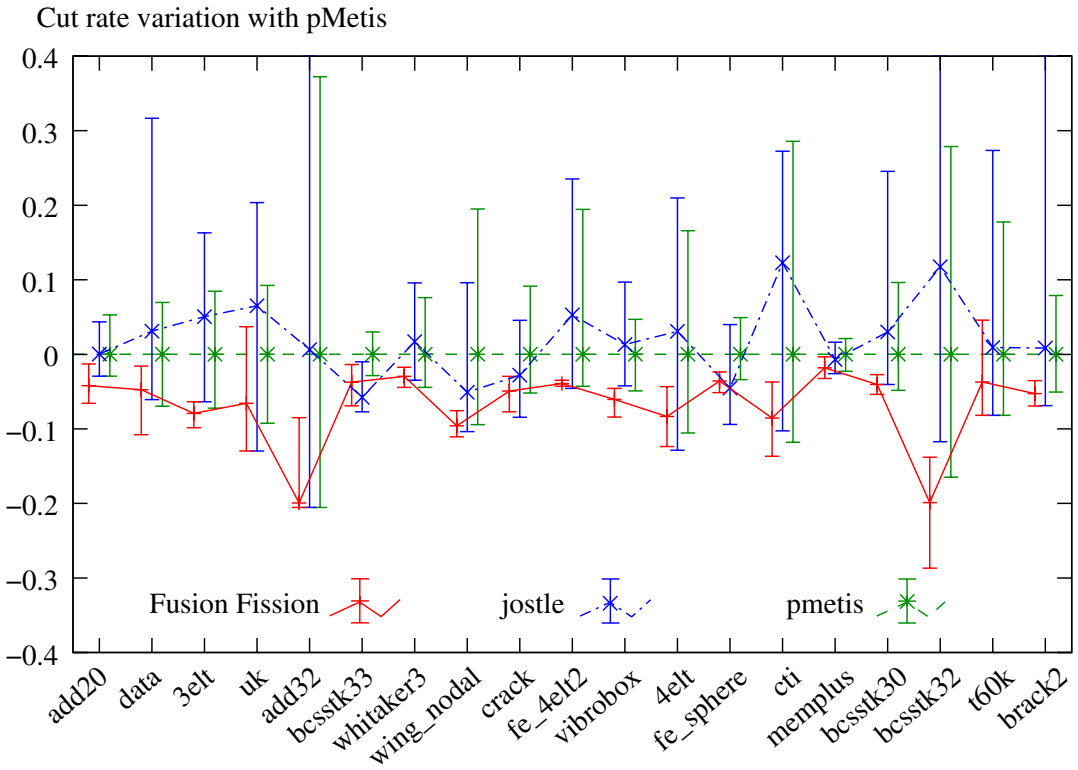

Fig. 5. Average and range Cut's rate variation results of 4 parts partitions returned by Fusion-Fission, Jostle and pMetis, with corresponding pMetis average Cut.

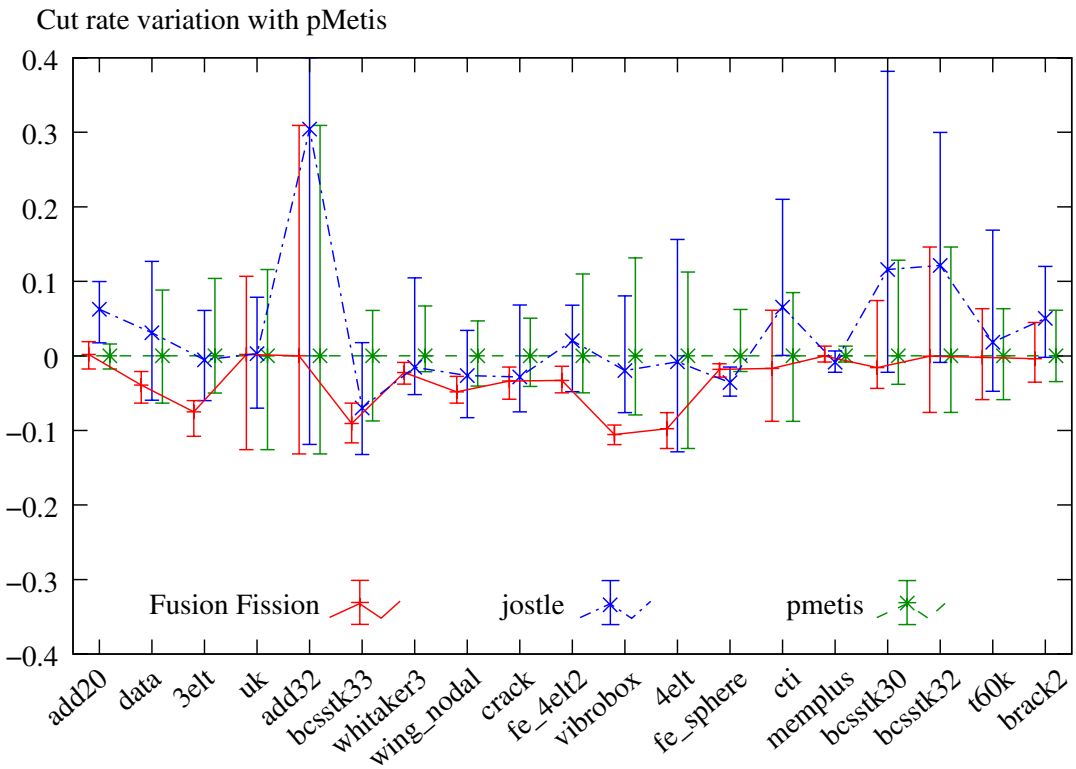

Fig. 6. Average and range Cut's rate variation results of 8 parts partitions returned by Fusion-Fission, Jostle and pMetis, with corresponding pMetis average Cut.

[11] Anders Hallgren. Restructuring european airspace: functional airspace blocks. *Skyway*, pages 20–22, autumn 2005.

[12] Bruce Hendrickson. Graph partitioning and parallel solvers: Has the emperor no clothes? (extended abstract). In *Proceedings of the Workshop on Parallel Algorithms for Irregularly Structured Problems*, pages 218–225, 1998.

[13] David S. Johnson, Cecilia R. Aragon, Lyle A. McGeoch, and Catherine Schevon. Optimization by simulated annealing: an experimental evaluation; part i, graph partitioning. *Operations Research Society of America*, 37(6):865–892, 1989.

[14] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal of Scientific Computing*, 20(1):359–392, 1998.

[15] George Karypis and Vipin Kumar. Multilevel k-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed Computing*, 48(1):96–129, 1998.

[16] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, 49(2):291–307, 1970.

[17] Aleix M. Martínez, Pradit Mittrapiyanuruk, and Avinash C. Kak. On combining graph-partitioning with non-parametric clustering for image segmentation. *Computer Vision and Image Understanding*, pages 72–85, 2004.

[18] Marina Meila and Jianbo Shi. Learning segmentation by random walks. In *Proceedings of the Neural Information Processing Systems*, pages 873–879, 2000.

[19] Franois Pellegrini. *Scotch and libScotch*. ENSEIRB - LaBRI, Universit de Bordeaux I, 4.0 edition, jan 2006.

[20] P. Perona and W. Freeman. A factorization approach to grouping. In *Proceedings of the European Conference on Computer Vision*, volume 1406 of *Lecture Notes in Computer Science*, 1998.

[21] Alex Pothen, Horst D. Simon, and Kang-Pu Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal on Matrix Analysis and Applications*, 11(3):430–452, 1990.

[22] J. Shi, S. Belongie, T. Leung, and J. Malik. Image and video segmentation: The normalized cut framework. In *Proceedings of the IEEE International Conference on Image Processing*, pages 943–947, 1998.

[23] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[24] Horst D. Simon and Shang-Hua Teng. How good is recursive bisection? *SIAM Journal on Scientific Computing*, 18(5):1436–1445, 1997.

[25] Alan J. Soper, Chris Walshaw, and M. Cross. A combined evolutionary search and multilevel optimisation approach to graph-partitioning. *Journal of Global Optimization*, 29:225–241, 2004.

[26] Chris Walshaw and M. Cross. Mesh partitioning: A multilevel balancing and refinement algorithm. *SIAM Journal on Scientific Computing*, 22:63–80, 2000.

[27] Song Wang and Jeffrey Mark Siskind. Image segmentation with minimum mean cut. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 517–524, 2001.

[28] Elie Yarack. An evaluation of move-based multi-way partitioning algorithms. In *Proceedings of the 2000 IEEE International Conference on Computer Design: VLSI in Computers & Processors*, page 363, 2000.

[29] Hongyuan Zha, Xiaofeng He, Chris H. Q. Ding, Ming Gu, and Horst D. Simon. Bipartite graph partitioning and data clustering. In *ACM Conference on Information and Knowledge Management*, pages 25–32, 2001.

[30] Ying Zhao and George Karypis. Empirical and theoretical comparisons of selected criterion functions for document clustering. *Machine Learning*, 55(3):311–331, 2004.