Fly4Arts : Evolutionary Digital Art with the Fly Algorithm

Zainab Ali Abbood[1] and Franck P. Vidal[1]

[1]School of Computer Science, Bangor University, UK, z.a.abbood@bangor.ac.uk, f.vidal@bangor.ac.uk

ABSTRACT. The aim of this study is to generate artistic images, such as digital mosaics, as an optimisation problem without the introduction of any *a priori* knowledge or constraint other than an input image. The usual practice to produce digital mosaic images heavily relies on Centroidal Voronoi diagrams. We demonstrate here that it can be modelled as an optimisation problem solved using a cooperative co-evolution strategy based on the Parisian evolution approach, the Fly algorithm. An individual is called a fly. Its aim of the algorithm is to optimise the position of infinitely small 3-D points (the flies). The Fly algorithm has been initially used in real-time stereo vision for robotics. It has also demonstrated promising results in image reconstruction for tomography. In this new application, a much more complex representation has been study. A fly is a tile. It has its own position, size, colour, and rotation angle. Our method takes advantage of graphics processing unit (GPU) to generate the images using the modern OpenGL Shading Language (GLSL) and Open Computing Language (OpenCL) to compute the difference between the input image and simulated image. Different types of tiles are implemented, some with transparency, to generate different visual effects, such as digital mosaic and spray paint. An online study with 41 participants has been conducted to compare some of our results with those generated using an open-source software for image manipulation. It demonstrates that our method leads to more visually appealing images.

KEYWORDS. Digital mosaic, Evolutionary art, Fly algorithm, Parisian evolution, cooperative co-evolution.

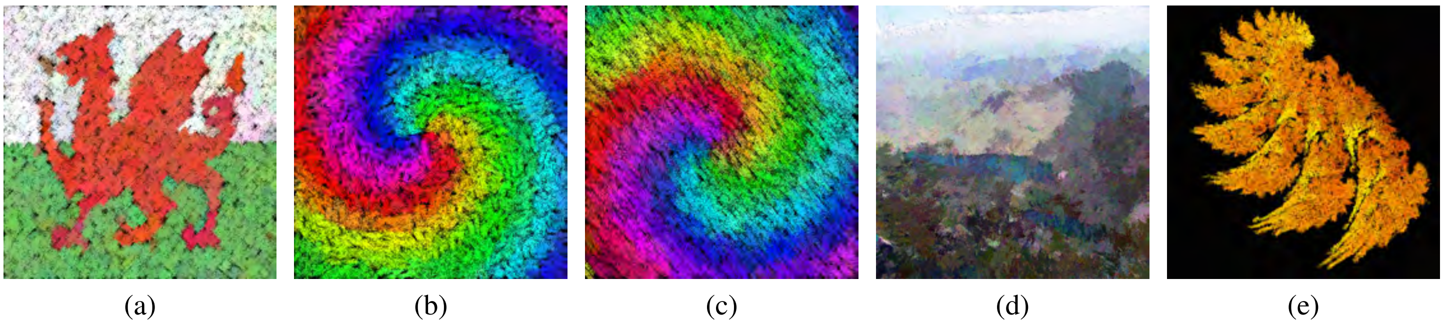


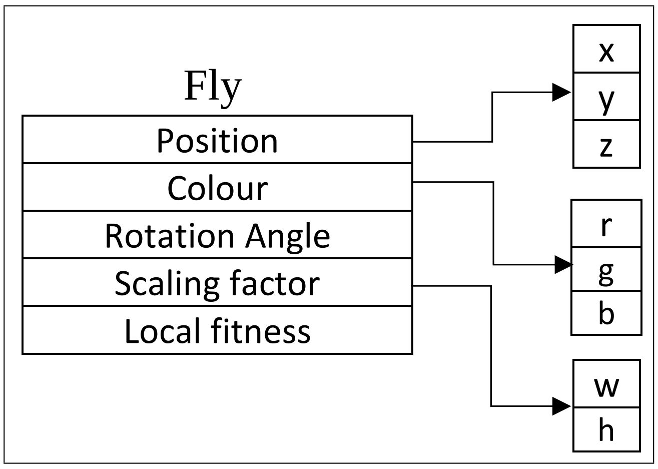
Figure 1. *Evolutionary art using the Fly algorithm : (a) Welsh flag, (b) a multicolour spiral, (c) another multicolour spiral, (d) view from Garnedd Ugain of Y Lliwedd in Snowdonia (Wales), and (e) from an image generated using ArtiE-Fract (courtesy of Dr Évelyne Lutton).*

1. Introduction

The problem of automatically translating an input image (e.g. a photograph) into digital art, such as mosaics, can be described using the notation below :

> *Given a rectangular region I^2 in the plane $\mathbb{R}^2$, a dataset of N tiles, a set of constraints, and a vector field $\phi(x, y)$ defined on that region, find N sites $P_i(x_i, y_i)$ in I^2 to place the N tiles, one at each site P_i, such that all tiles are disjoint, the area they cover is maximised and the constraints are verified as much as possible.*

The most common techniques to express (adapt) it to generate digital art depend on Centroidal Voronoi (CV) diagrams (see Section on Previous Work in [1] for details). In this paper, we solve it as an optimisation problem using a cooperative co-evolution approach based on the Parisian evolution strategy, the Fly algorithm. Its main feature is to consider each fly as a part of the problem's solution. Here the



(a) *Structure of the fly data.*

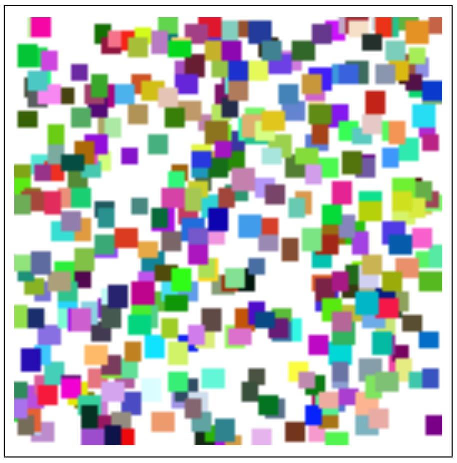

(b) *Random initial population.*

Figure 2. *Structure of an individual fly and the whole population.*

goal is to find the best set of tiles and how to assemble them onto a rectangular region to approximate a colour image. Figure 1 shows reconstructed images using such sets of tiles. Each tile is characterised by 4 features : position, colour, scale and rotation angle. Our algorithm is partly implemented on GPU to speed up computations : i) digital images are generated using GLSL, and ii) the fitness function is computed using GLSL and OpenCL. The evolutionary algorithm is executed on the central processing unit (CPU).

This article is organised as follows. Section 2 explains how to adapt the Fly algorithm to generate digital art. The following section presents our results. Several images have been created using different versions of the algorithm to generate various visual effects. This section presents an online study in which 41 participants had to judge the visual quality of individual images. Some were generated with our approach, others were corresponding images generated with a famous open-source software. Concluding remarks are given in the last section.

2. Evolutionary digital arts

To solve the 2-D digital art reconstruction problem, we follow the same main principles as in 3-D using the Fly algorithm [2]. The individuals still correspond to extremely simple primitives : The flies. In positron emission tomography (PET) reconstruction, flies corresponded to 3-D points only. Here the structure of flies is more complex to correspond to rectangular tiles. Therefore, each fly is a vector of 9 elements (see Fig.2a) :

Position is a 3D point with coordinates (x, y, z), which are randomly generated between 0 and $width - 1$, 0 and $height - 1$, and -1 and 1 respectively (with $width$ and $height$ the number of pixels in the image along the x- and y-axes).

Colour has three components (r, g, b) (for red, green and blue), which are randomly generated between 0 and 1. This is to ensure diversity at the start of the optimisation.

Rotation Angle is randomly generated between 0 and 360.

Scaling factor has two components (w, h), which control the size of the tile along its horizontal and vertical axes.

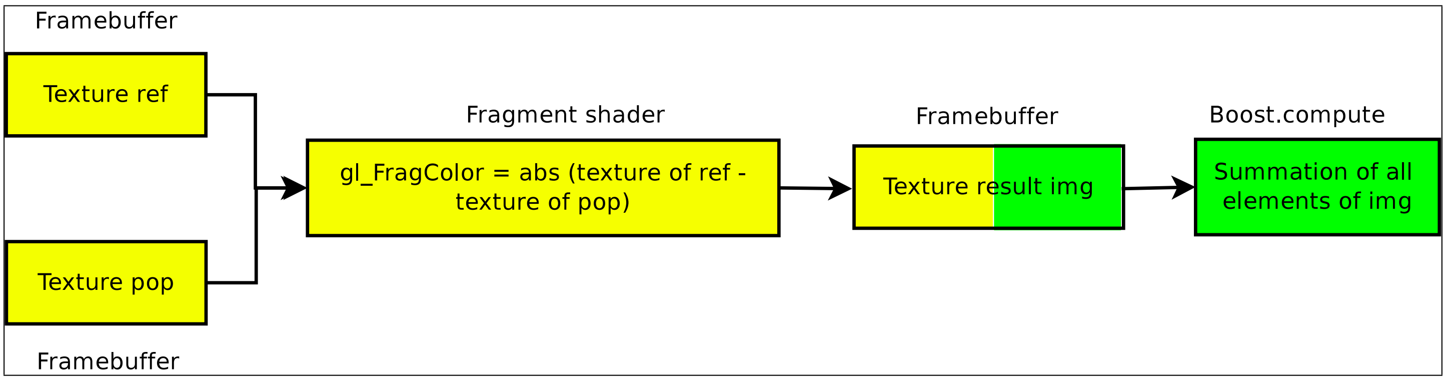


Figure 3. *Computation of the marginal fitness on GPU for two images using GLSL and OpenCL (in yellow and green boxes respectively).*

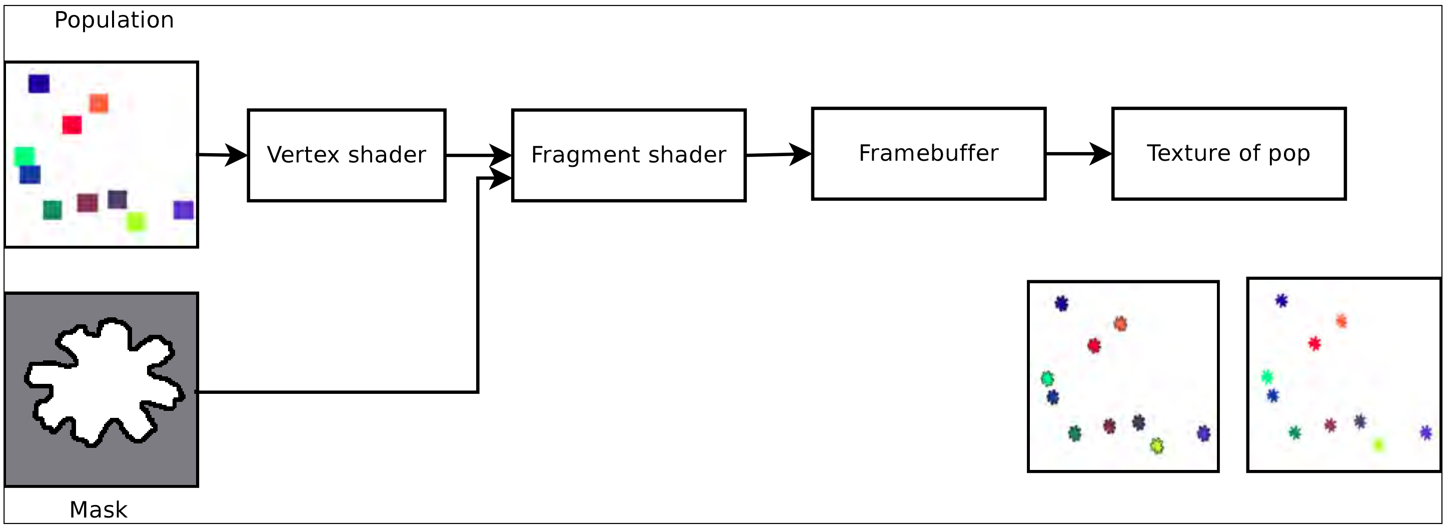


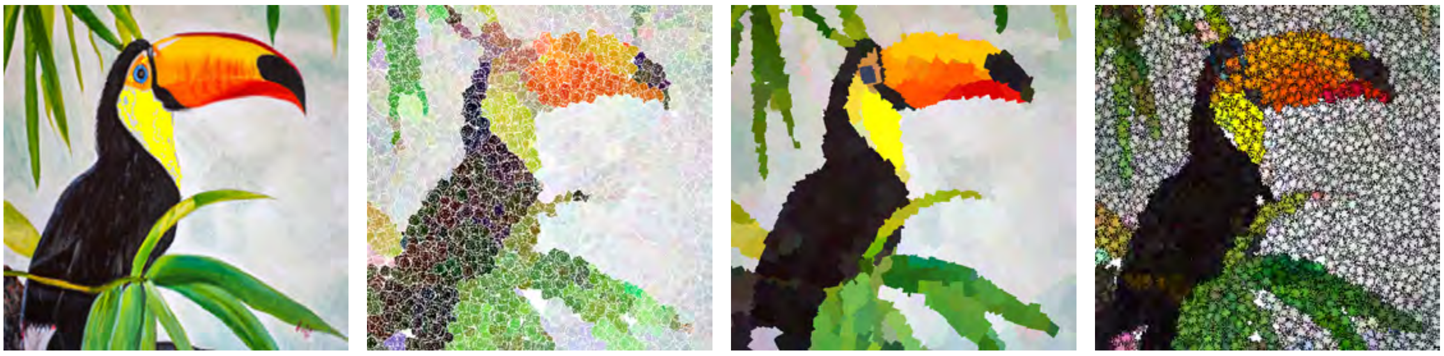
Figure 4. *Generation of pop using different masks and GLSL fragment shaders.*

Our algorithm follows a steady-state evolutionary strategy. At each iteration of the optimisation process, a bad fly is selected for death and replaced with another one using genetic operators. The algorithm starts to create an image based on a random initial population. An example of reconstructed image is shown in Figure 2b. The algorithm's aim is to minimise the difference between the reconstructed image (pop) and the input image (ref). This difference is called *global fitness* and it quantifies how good the whole population is. During the evolutionary process, the flies are selected based on their own *local fitness* and genetic operators applied (mostly mutation and new blood).

Global fitness : We assess the whole population by comparing the input image with the simulated image. We use the sum of absolute error (SAE) (also known as Manhattan distance) to quantify the error between ref and pop :

$$\mathrm{SAE}(pop, ref) = \sum_{i=0}^{i<width} \sum_{j=0}^{j<height} |ref(i,j) - pop(i,j)| \qquad [1]$$

Computing the SAE in this application is time consuming on a CPU. Therefore, all the computations to generate images are performed on a GPU using the GLSL (see Figure 3). pop is generated offline using a framebuffer object (FBO). Figure 4 shows the process of fitting a tile with certain mask (it can produce different visual effects by using different masks and fragment shaders). Images (including pop and ref) are stored using 2-D OpenGL textures. The textures are then pas-



(a) *Reference image.* (b) *Image reconstructed using the set of stripes mask.* (c) *Image reconstructed using the square mask.* (d) *Image reconstructed using the flower mask.*

Figure 5. *Examples of 2-D Reconstruction using the Fly algorithm. The original image is from the Open Images Dataset (*`https://github.com/openimages/dataset`*) under CC BY 4.0 license.*

sed to a GLSL shader program to compute the pixel-wise absolute error between ref and pop. The sum is also performed on the GPU, but using the OpenCL implementation of the reduction operator supplied by Boost.Compute [3, 4]. It provides the SAE in an effective manner.

Local fitness : Our implementation looks for 'bad' individuals to replace them by 'good' ones. During the selection process based on the *Threshold selection* operator [5], we measure how good or bad the contribution of Fly i is. It relies on the SAE metrics with the leave-one-out cross-validation method, which is why the local fitness is also called *marginal fitness*, F_m :

$$F_m(i) = \mathrm{SAE}(pop - \{i\}, ref) - \mathrm{SAE}(pop, ref) \quad [2]$$

The population is assessed with and without Fly i. Therefore, if the SAE is greater with Fly i than without, then Fly i has a negative effect on the performance of the population : the fly could be killed. If the error is smaller with Fly i than without, then Fly i has a positive impact on the performance of the population : the fly is a good candidate for reproduction. $\mathrm{SAE}(pop - \{i\}, ref)$ is computed in a similar manner as $\mathrm{SAE}(pop, ref)$.

3. Results

Figure 5 show some results of the same image (Toucan) using different masks. More results are available as videos on YouTube at `http://tinyurl.com/ho5kfvb`. Our algorithm is relatively flexible as it can adjust to different schemes such as :

— Different image sizes ;
— Different colour quantisations ;
— Different shapes of tiles : flower, set of lines and square (see Figure 6).
— With or without restart mechanism.

To demonstrate the usefulness of our approach, a comparison study between the proposed algorithm and the GNU Image Manipulation Program (GIMP) has been performed using three test images (Glasses, Bird and Woman) and three types of masks (rectangles, flower and stripes) (see Figure 6). It can be visually observed in Figure 7 that our method leads to better reconstructions in term of edges and colours (brightness). We conducted an online survey to rate the corresponding 18 images (3×3 created using our method, 3×3 using GIMP). The images are presented in a random order to the participant during

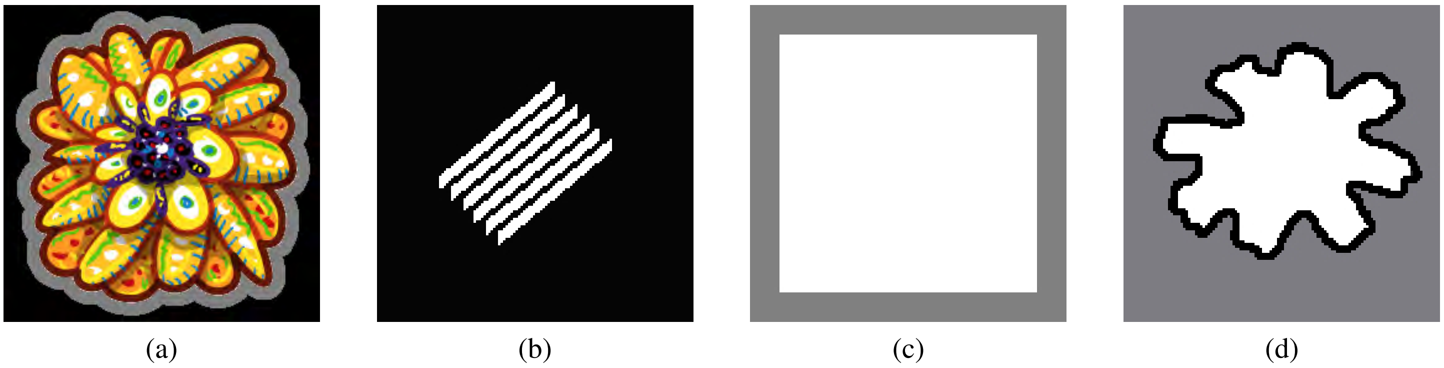


Figure 6. *Examples of tile templates.*

Tableau 1. *Absolute category rating of the images of Figure 7*

Rating	Label	Count of proposed method	Count of GIMP
5	Excellent	32	7
4	Good	86	50
3	Fair	110	78
2	Poor	71	125
1	Bad	19	59

the survey. For each image, we asked the participants to rate its quality in terms of visual appearance. The rating scale used during the evaluation is called *Absolute Category Rating* scale. It maps ratings between bad and excellent to numbers between 1 and 5. Table 1 illustrates how many times each label was selected for images of our method and for GIMP. The mean opinion score (MOS) metrics has been used to a quantitative evaluation of the results. For each image, MOS is calculated as follows :

$$MOS = \frac{\sum_{n=0}^{N} R_n}{N} \qquad [1]$$

where R are the individual ratings for a given stimulus by N participants. Figure 7 shows the MOS values for each image. The MOS values of the images of our method are systematically higher than the corresponding ones of GIMP. The MOS values for all images are also ranked. It shows that the images of our method are mostly in the beginning of the list (1, 2, 3, 4, 6, 8 and 9). The images of GIMP are in the end of the list (11, 12, 13, 14, 15, 16, 17 and 18). There is one exception : the glass image of GIMP with the flower mask is #5. However, the MOS value of that image is 3.1, which is less than the same image of proposed method (3.4).

4. Conclusion

This article showed that the Fly algorithm can be adapted to the field of digital arts for creating visual effects on an image in a fully-automatic fashion. Before the algorithm was used in PET reconstruction. We moved from a simple structure of fly to a more complex one with a lot more properties than ever. Our implementation takes advantage of real-time computer graphics rendering techniques to generate the image data and GPU computing to calculate the fitness functions. An online comparison study (user evaluation survey) has been performed, using 18 images, amongst which 9 were generated by our proposed method and 9 using GIMP. According to the participants, our proposed method produced better images

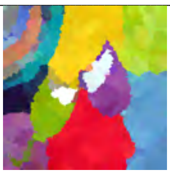
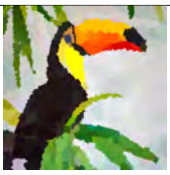
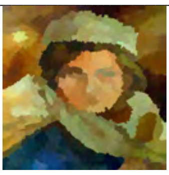
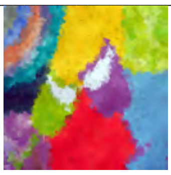
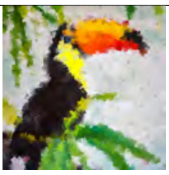
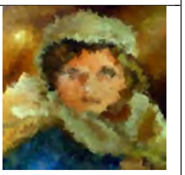
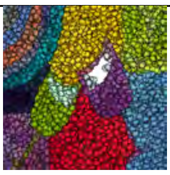
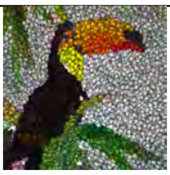
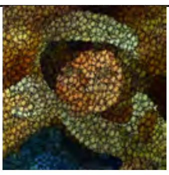
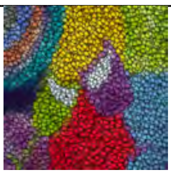
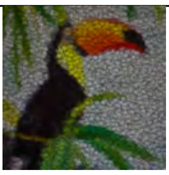
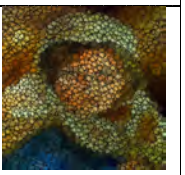
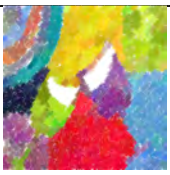
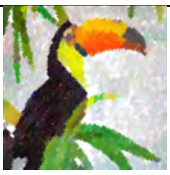
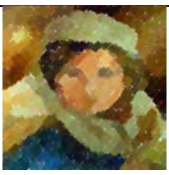
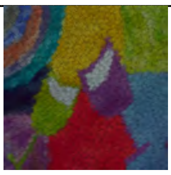
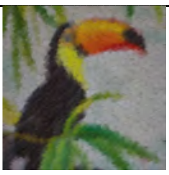
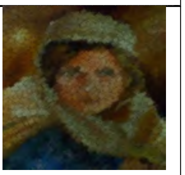
Masks	Images of proposed method			Images of GIMP		
	Glasses	Bird	Woman	Glasses	Bird	Woman
Square						
MOS	2.9	3.3	2.8	2.5	2.6	2.5
Rank	6	4	9	12	11	13
Flower						
MOS	3.4	3.6	2.9	3.1	2.4	2.5
Rank	3	1	6	5	15	14
Stripes						
MOS	2.8	3.4	2.8	2.1	2.0	2.4
Rank	9	2	8	17	18	16

Figure 7. *Images used in the online survey. The woman image (Fatima) is from the artist Lubna Ashrafis. Other test images are from the Open Images Dataset (`https://github.com/openimages/dataset`) under CC BY 4.0 license.*

in terms of visual appearance. Future work will include improving the performance of the algorithm to use progressive multi-resolution based on the mitosis operator initially developed for PET reconstruction using the fly algorithm, and the development of a filter (plugin) for GIMP.

Bibliographie

Ali Abbood, Z., Amlal, O., Vidal, F.P. : Evolutionary art using the Fly algorithm. In : Applications of Evolutionary Computation. Volume 10199 of Lecture Notes in Computer Science., Springer, Heidelberg (April 2017) 455–470

Ali Abbood, Z., Lavauzelle, J., Lutton, E., Rocchisani, J.M., Louchet, J., Vidal, F.P. : Voxelisation in the 3-D Fly algorithm for PET. Swarm and Evolutionary Computation (2017) In press.

Lutz, K. : Boost.Compute. `http://boostorg.github.io/compute/` Accessed : 2016-10-26.

Gaster, B., Howes, L., Kaeli, D.R., Mistry, P., Schaa, D. : Heterogeneous Computing with OpenCL. 1st edn. Morgan Kaufmann (2011)

Vidal, F.P., Louchet, J., Rocchisani, J.M., Lutton, É. : New genetic operators in the Fly algorithm : Application to medical PET image reconstruction. In : Applications of Evolutionary Computation : EvoApplicatons 2010. (2010) 292–301