Product/Process Configuration Evolutionary Optimization: A Multiobjective Clustering in Order to Reduce Inconsistencies During Crossover

Paul Pitiot, Michel Aldanondo, Élise Vareilles, Paul Gaborit

Product/Process Configuration Evolutionary Optimization: A Multiobjective Clustering in Order to Reduce Inconsistencies During Crossover

P. Pitiot[1,2], M. Aldanondo[1], E. Vareilles[1], P. Gaborit[1]
[1]Université de Toulouse - IMT Mines Albi-CGI, Albi, France
[2]Institut d'Ingénierie Informatique de Limoges, Rodez, France
(michel.aldanondo@mines-albi.fr, elise.vareilles@mines-albi.fr, p.pitiot@aveyron.cci.fr)

Abstract - **Concurrent configuration of a product and its associated production process is a challenging problem in customer/supplier relations dealing with configurable products. Search for optimized solutions that respect customer's needs and constraints of the problem in a multiobjective context is a particularly difficult task. Constraints Filtering Based Evolutionary Algorithm (CFB-EA) [1] proposes an original way to integrate constraints satisfaction in optimization thanks to a constraints filtering engine. CFB-EA tries to mix solutions randomly selected in order to improve them but leads to many incompatibility occurrences which are time consuming. We propose in this article a dedicated multiobjective clustering algorithm that reduces incompatibilities occurrences and improve the selection of solutions for crossover operator.**

Keywords - **product configuration; process configuration; configuration optimization; evolutionary algorithms; clustering**

I. INTRODUCTION

In the manufacturing industry, the concept of mass customization has established itself as an indispensable lever for gaining market share. In order to develop and implement mass customization, many companies use configuration software [2]. Configuration software enables companies to propose to their customers customized products from a huge set of variants and options of products [3]. If all configuration software can assist the supplier during the configuration of the product, only some of them do the same for the associated production and/or delivery process. As product and process configurations are frequently interdependent, some authors [4][5][6] have proposed combining them in a single problem called 'Concurrent Product and Process Configuration' (CPPC). The configuration software is used to confront customer's requirements and supplier needs with the CPPC model. Processing and optimizing CPPC problems is a major issue for companies producing technical products or systems either in business to business or business to customer situations. In addition to a solution that respects constraints and its requirements, the user (either customer or supplier) is also interested in an optimized response according to multiple criteria as for example cost, cycle time sustainability, quality or technical performance (in this study, only the first two ones are considered).

This result in a multicriteria optimization problem called O-CPPC. This problem is particularly difficult because of the huge number of variants of the product and the process, as well as their relationships and their impact on various objectives.

To face this problem, we propose in previous studies [1], a decision aiding process in two steps presented in Figure 1. Step1: interactive configuration and planning which processes non-negotiable requirements and provides a first solution space reduction. Step 2: response optimization which take into account constraints and provides a Pareto front shown to the customer for a final solution selection. This study is mainly concerned by improvement of this second step.

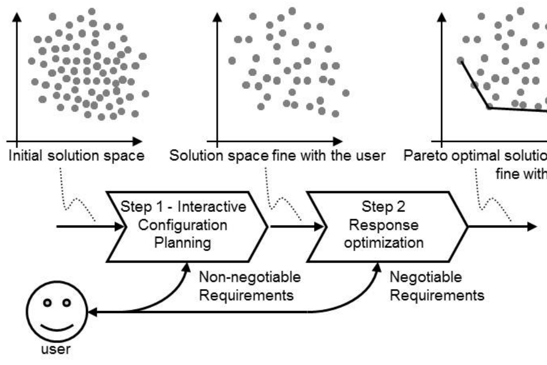
Initial solution space
Solution space fine with the user
Pareto optimal solutions space fine with the user
Step 1 - Interactive Configuration Planning
Step 2 Response optimization
Non-negotiable Requirements
Negotiable Requirements
user

Figure 1 – Decision aiding process

CFB-EA, presented in Figure 2, was proposed to achieve this multiobjective and constrained optimization step. There are various ways to integrate constraints handling in evolutionary algorithms [7]. The specificity of CFB-EA relies on an interaction with a filtering engine that maintain feasibility of solutions during their construction (initialization step) or modification (crossover and mutation step). CFB-EA algorithm is based on an adapted version of SPEA2 [8] that integrate constraints handling by the filtering engine.

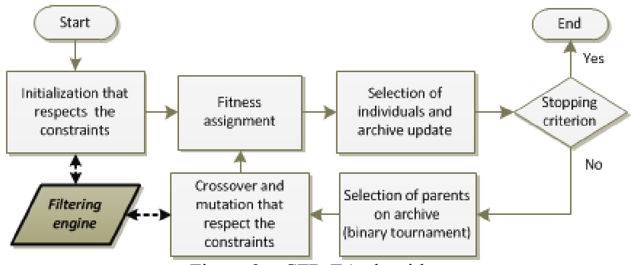
Start
End
Yes
Initialization that respects the constraints
Fitness assignment
Selection of individuals and archive update
Stopping criterion
No
Filtering engine
Crossover and mutation that respect the constraints
Selection of parents on archive (binary tournament)

Figure 2 – CFB-EA algorithm

The main ideas of SPEA2 are: a) the evaluation of a solution takes Pareto-dominance and the local density of solutions into account, b) a set of the best and most diversified solutions are preserved in a separate archive, c) a binary tournament in the archive is used to select parents for the next generation.

Six parameters have to be set: size of archive, size of population, number of generations or any stopping criterion, crossover probability for individual selection, mutation probabilities for individual and gene selections.

In the current version, the selection of parents for a crossover is achieved randomly. But in a constrained context, mixing very different individuals could be difficult. It leads to incompatibilities and thus a waste of computation time with many backtracks. The key idea put forward in this article is to retrain the parent selection to close individuals in the search space. This idea is not new and corresponds to the concept of restricted crossover or matting restriction radius (i.e. the maximal distance between the two parents) in unconstrained optimization [9]. The specificity of our proposition relies on the constrained and multiobjective context. We thus make the assumption than closed individuals are more compatible than distant ones. Such process could reduce waste of computation time due to incompatibilities. Using an absolute distance measure in the search space would be tricky to define and to set. Clustering analysis is an interesting unsupervised way to avoid using a distance measure. We thus propose an original multiobjective clustering of the solutions of the archive, then a selection of parents in close clusters for the crossover.

In next sections, a formal definition of the problem and a description of actual crossover operator are recalled. Then the proposed improvement and the associated clustering algorithm are presented. An experimental validation is finally exposed.

II. PROBLEM FORMULATION AND RELATED WORKS

A. Formal definition of O-CPPC problem

Product configuration can be defined by selection of a specific or customized product (through a set of properties, sub-assemblies or bill of materials, etc.) from a generic product or a product family, while taking into account specific customer requirements [10]. The same kind of reasoning can be considered for process configuration which consists in the selection of a specific production plan (set of operations, resources to be used, etc...) from some kind of a generic process plan while respecting process characteristics. Those two configuration problem and their coupling can be considered as a constraint satisfaction problem (CSP). This means that we could represent this CPPC problem by a set of variables with associated domains linked by constraints. But we could also add to this representation variables needed to evaluate configurations. Typically, customers are at least interested in knowing price and delivery date of his configured product. That leads us to define the O-CPPC problem as the search for product and process configuration that meet customer product/process requirements but also evaluation expectations.

This can be formulated using O-CSP (optimization constraint satisfaction problem) modeling framework. An O-CSP is defined by the quadruplet <V, D, C, f > where V is the set of decision variables, D the set of domains linked to the variables of V, C the set of configuration constraints on variables of V and f the multi-valued fitness function:

- The set V gathers the product and process decision variables but also some evaluation variables. Evaluation variables (cost and duration in our case) allow calculating fitness of a solution. Decision variables are all symbolic or at least discrete, while evaluation variables are numerical.
- The set C gathers all configuration constraints (mainly with tables of compatibility between values in domains of linked variables).
- The fitness function is defined by various numerical constraints and tables of compatibility that allows calculating the objective values according to choices made on decision variables. On the product side, each decision variable (choice of components or functionalities) is linked to an evaluation variable by a table of compatibility that defines the cost of components or functionalities. Then a numerical constraint, the sum of all elementary costs, provides the global product cost. On the process side, for each operation, a table of compatibility allows to calculate the duration and the cost of the operation. Then total cycle time of the process is computed using numerical constraints according to precedence between operations. The process cost is calculated by a numerical constraint that sums all operation costs. Product and process cost are added to constitute the total cost of the configuration.

A strong specificity of this kind of optimization problem is that the solution space is large. Another specificity lies in the fact that the shape of the solution space is not continuous and, in most cases, shows many singularities. Furthermore, the multi-criteria problem and the need for Pareto optimal results are also strong problem expectations. These points explain why most of the articles published on this subject consider genetic or evolutionary approaches to deal with this problem [11][12][13][14].

B. Crossover operator in CFB-EA

A detailed description of CFB-EA could be founded in [1]. We focus in this paper on crossover operator. The crossover consists in mixing chromosome of two solutions (parents) to get two new solutions (child) potentially better than previous ones. It starts by the selection of two parents in the matting pool (the set of individuals selected to be parents of next generation thanks to a binary tournament). This parent selection is achieved randomly. A first parent is selected then a second one different from the first one (a good solution may appear several times in the matting pool).

Once parents are selected, an instance in filtering engine is initiated with the common part of parents then duplicated to achieve separately crossover of each children. A key point concerning time consumption in filtering engine is that first filters on an instance are by far the most costly in computation time. The more the common part is larger, the quicker will be the next filtering.

Crossover by itself consists in a uniform crossover of the remaining genes (genes different between parents). This means that each remaining gene have a probability of 50% to be selected between parents to get a child. Remaining genes are selected randomly one by one and, according to the gene crossover probability, their values are exchanged between parents. After each instantiation, filtering engine is called to check feasibility of resulting individuals. If a domain of remaining variables become empty, the individual is unfeasible. A backtrack on previous choices is then launched to restore feasibility. If a domain of remaining variables is reduced to only one value, this value is automatically selected in order to preserve consistent genes combinations of a parent and reduce the number of backtrack (a crossover on this gene would be impossible with respect to the instantiation of the previous one). This last process reduces crossover possibilities and leads to more useless crossover (i.e. a crossover that leads to obtain the same individual as one of his parents). Even if there is more useless crossover, the saving of computation time due to the reduction of backtrack allows to generate more individuals and leads to better results.

C. Clustering and evolutionary algorithm

Cluster analysis is the task of grouping a set of objects in groups (cluster) in the way that each object in the same cluster are more similar to each other than to those in other clusters [15]. Clustering has been successfully applied in various engineering and scientific disciplines such as biology, medicine, machine learning, pattern recognition, image analysis and data mining. Selection of a clustering algorithm depends on the number of solutions to sort and the required precision of clustering. The reader can consult [16] for a detailed survey of clustering algorithms. Here we have a small number of solutions (few hundreds) and a sub-optimal clustering is enough to get close individuals quickly. The simplest and most popular clustering algorithm is the K-means algorithm. The k-means algorithm has been used to regulate selection process in an EA named KGA [17] and to adjust the probabilities of crossover and mutation in [18]. In multiobjective context, Zhang et al. [19] propose a decision variable clustering method that divides the decision variables into two clusters based on the features of each variable. The decision variable clustering method adopts the k-means method to divide the decision variables into two types: convergence-related variables and diversity-related variables. Jointly to our work, [20] proposes a self-adaptive mating restriction strategy. It differs by the way clusters are formed and used.

III. PROPOSITION OF A RESTRAINED CROSSOVER

To reduce unsatisfying or difficult crossovers, the idea is to restrain crossover to parents that belong to close areas. Close parents should have more similarities and thus lead to more successful and quick crossovers. The drawback could be a lack of diversity that leads to suboptimal individuals or a slower improvement by limiting the diffusion of performing combinations in population. In constrained optimization context, another critical issue is to reduce the number of backtrack/repair during construction of individuals.

Our specificities rely on genetic operator addressed (only crossover operator) and the multi-objective problem processing. This last one imposes a normalization of objectives but also leads to an original way to cluster individuals. Indeed, dominated individuals in archive are not suitable to constitute interesting clusters. Thus we combine a normalization of individuals performance using the standard score (Z-score) and a clustering of individuals using k-means algorithm only on Pareto front individuals of the archive.

The Z-score calculation allows computing distances between individuals properly by eliminating the differences of scales between dimensions. Z-score of an individual on a dimension *i* is obtained by subtracting the population mean on this dimension from individual raw score and then dividing the difference by the population standard deviation as defined in equation 1. At the beginning of the clustering process, Z-score is thus calculated for every individual of archive.

$$Z_i = \frac{x_i - \mu_i}{\sigma_i} \tag{1}$$

Where i is the dimension evaluated, x_i is the performance of evaluated individual on dimension i, μ_i is the mean of the population on dimension i and σ_i is the standard deviation of the population on dimension i.

Once the z-scores are calculated on each individual of the archive, the k-means algorithm is launched. Parameters of k-means algorithm are *n* the number of clusters and *MaxIter* the maximal number of iterations. The specificities of our implementation are:

- the use of Z-score to calculate distances between individuals,
- clustering only on Pareto Front individuals in order to avoid useless Pareto-dominated clusters
- Initialization of clusters is achieved by sorting Pareto front individuals according to one criterion and partitioning in *n* clusters. This initialization of cluster is really effective because the algorithm converges in very few iterations (mostly in less than 5 iterations).

After initialization step, the algorithm follows classical steps: calculate centroids of each cluster, assign individuals to the nearest cluster and loop until there is no more change or *MaxIter* is reach (it never appends).

After this step, all remaining individuals of the archive (i.e. dominated ones) are assigned to the nearest cluster. Figure 3 illustrate two cases of clustering process results: for an archive at the beginning of optimization process (on left with spread individuals) and at the end (on right with grouped individuals).

Finally, the initial parent selection of CFB-EA is modified as follow. Individuals are selected for crossover according to the crossover probability. For each selected individual, another parent is selected in archive in the same or in the close clusters. In this first version and to avoid numerous parameters, second parent selection is limited to the nearest clusters (i.e. an individual that belongs to cluster *i* could by crossed with individuals that belongs to cluster *i-1*, *i* or *i+1*).

Thus the only new parameter is the number of clusters. To avoid suboptimal convergence and ensure propagation of performing combinations in the population, clusters must not be too small. If they are too large, benefits of restrained crossover could be wasted. A future improvement could be to automatically set this parameter according to the mean number of individuals in clusters with regard to the size of archive.

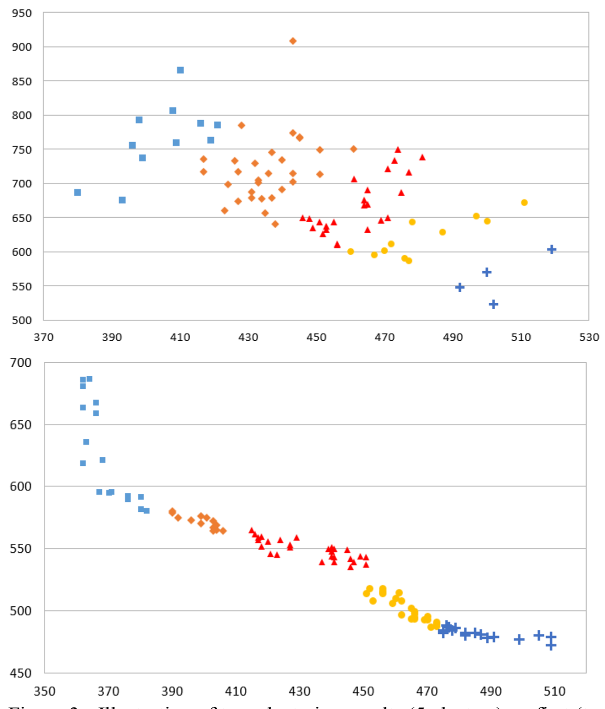

Figure 3 - Illustration of two clustering results (5 clusters) on first (top) and on last archive (bottom) of one run.

IV. EXPERIMENTS

A. Evaluation metrics and problem instance tested

To evaluate proposed approach, we propose to compute and to follow, with respect to the computation time, the evolution of the hypervolume (HV) metric defined in [21]. This measures the hypervolume dominated by a set of individuals. HV is maximum when the solutions are the most diversified on the Pareto front. Each result is the average of 5 runs. Average HV (undimensional) and average computation time (in seconds) and their respective relative standard deviation (RSD) are provided. To evaluate evolution of HV, we also compute time needed to get 99.9% and 99% of final HV value. Notice that the time consumption corresponds to one CPU core processing time and, as filtering engine could be parallelized, it could be divided by the number of CPU core used to get the real computation time.

These experiments are made on a reference problem that considers 30 configuration variables (24 product description variables and 6 process description variables). 26 configuration constraints link between 2 and 4 configuration variables and, for each constraint, the ratio of feasible tuples is between 0.01 and 0.6. In order to avoid case dependency, the used problem is drawn from an analysis of CPPC problems proposed in [22]. This analysis is based on the notions of generic product modules, generic process operations, configuration and constraint patterns. It allows various representative CPPC problems to be generated. In this experiment, we use a model that represents a platform architecture product model.

B. Impact of restricted crossover and parameter tuning

Figure 4 shows the results achieved with classical CFB-EA (continuous line) and with proposed restricted crossover (dotted line). Here, archive is divided in 5 clusters as illustrated on figure 3. Associated values could be founded in first two line of Table 1.

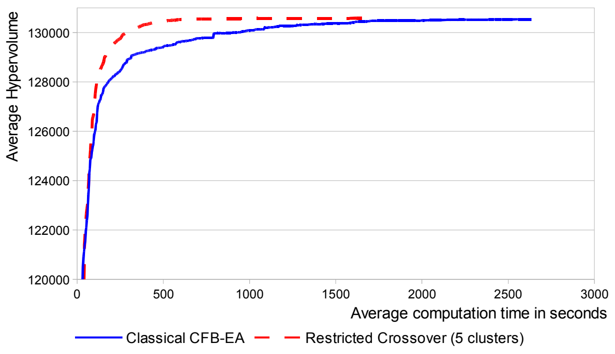

Figure 4 – Effect of restricted crossover

Effect of restricted crossover is really significant. 33% less computation time is needed to get the final HV value and 68% to get 99.9% of final HV. CFB-EA with restricted crossover found the best value at every run (RSD of HV is 0 %) in 1695 seconds. However, there is a strong dispersal (RSD of time is 28.5%).

Detailed analysis shows that the number of backtracks is reduced by 34%. On the other hand, the number of useless crossovers strongly increases (37% of useless crossover against 23% without restricted crossover).

	HV_{final}				99,9% of HV_{final}	
	time (sec.)	RSD time in %	HV	RSD HV in %	time	RSD time in %
CFB-EA	2556	23,526	130530	0,08	1592	2,353
RC 5 clust.	1696	28,579	130587	0	505	0,642
RC 3 clust.	2170	20,884	130575	0,019	830	1,655
RC 8 clust.	1972	44,443	130583	0,007	1138	1,700

Table 1 – Detailed results

We also analyze the impact of the number of clusters. Best results are obtained with 5 clusters. With 3 clusters, behavior of algorithm is hardly better than CFB-EA without restricted crossover. There is only 8% less backtracks. With 8 clusters, the number of backtracks is reduce by 38%. Nevertheless, number of useless crossover also strongly increase to 46% of useless crossover and leads to poorer results (16% / 125% more computation time to get final HV / 99.9% of final HV with an RSD on time of 44% / 1.7%).

V. CONCLUSION

In this paper, we present an effectiveness improvement of CFB-EA thanks to a restrained crossover. The k-means clustering algorithm was adapted to the multiobjective context in an original and simple way. It could be used with others multiobjective EA. First experimental results clearly show significant computation time reductions with an order of magnitude of 33%. This must be confirmed with others models experimentations and compared with other existing clustering approaches. In our to-do list, an automatic setting of the number of cluster is also interesting to study.

REFERENCES

[1] P. Pitiot, M. Aldanondo, E. Vareilles, P. Gaborit, M. Djefel, and S. Carbonnel. "Concurrent Product Configuration and Process Planning, towards an Approach Combining Interactivity and Optimality." in *International Journal of Production Research*, vol. 51 (2) pp 524–541, 2013.

[2] L. Hotz, A. Felfernig, A. Gunter and J. Tiihonen. "A short History of Configuration Technologies", in chapter 2 of *Knowledge-based Configuration: From Research to Business Cases*, Elsevier/Kaufmann, pp 9-17. 2014.

[3]. L. Wang, S. Sheng Zhong and Y. Jian Zhang. "Process Configuration based on generative constraint satisfaction problem", *Journal of Intelligence Manufacturing*. Vol. 28, Issue 4, pp 945-952, 2015.

[4] D. Campagna and A. Formisano. "Product and Production Process Modeling and Configuration." in *Fundamenta Informaticae* 124 (4) pp. 403–425, 2013.

[5] L. Zhang, E. Vareilles, and M. Aldanondo. "Generic Bill of Functions, Materials, and Operations for SAP 2 Configuration", in *International Journal of Production Research* 51 (2) pp. 465–478, 2013.

[6] H. Karppinen, K. Seppänenand, and J. Huiskonen, "Identifying Product and Process Configuration Requirements in a Decentralised Service Delivery System." in *International Journal of Services and Operations Management* 17 (3) pp 294–310, 2014.

[7] E. Mezura-Montes and C. Coello Coello, "Constraint-handling in Nature-inspired Numerical Optimization: Past, Present and Future.", in *Swarm and Evolutionary Computation* 1 (4) ,pp 173–194, 2011.

[8] E. Zitzler, M. Laumanns and L. Thiele. "SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization", *Evol. Methods for Design, Optimisation and Control*, pp 95-100, CIMNE, 2002.

[9] A Petrowski and S. Ben Hamida, "Evolutionary Algorithms", in *Metaheuristics*, ed. P Siarry, Springer, chap 5. pp 115-173, 2016.

[10] M, Aldanondo, E, Vareilles, "Configuration for mass customization: how to extend product configuration towards requirements and process configuration", Journal of Intelligent Manufacturing, vol, 19 n° 5, pp 521-535, 2008.

[11] L. Li, L. Chen, Z. Huang and Y. Zhong, "Product configuration optimization using a multiobjective GA", in *I.J. of Adv. Manufacturing Technology* vol. 30, pp 20-29, 2006.

[12] W. Wei, W. Fan and Z. Li. "Multi-objective optimization and evaluation method of modular product configuration design". *International Journal of Advanced Manufacturing Technology*. Volume: 75, Issue: 9-12, pp 1527-1536, 2014.

[13] Z. Xu and M Liang, "Concurrent optimization of product module selection and assembly line configuration: A multi-objective approach". *Journal of manufacturing science and engineering*, Vol.127 (4), pp 875-884, 2005.

[14] C. Zhou, Z. Lin and C. Liu, "Customer-driven product configuration optimization for assemble-to-order manufacturing enterprises", *International Journal of Advanced Manufacturing Technology*. Volume: 38, Issue: 1-2, pp 185-194, 2008.

[15] Z.-Y Li, J.-H. Yi, G.-G. Wang, "A new swarm intelligence approach for clustering based on krill herd with elitism strategy" in *Algorithms*, vol. 8, pp 951–964, 2015.

[16] S.K. Popat, M. Emmanuel, "Review and comparative study of clustering techniques", *in Int. J. Computing Science Information Technologies*, vol. 5, pp 805–812, 2014.

[17] A. Chehouri, R. Younes, J. Khoder, J. Perron and A. Ilinca, "A Selection Process for Genetic Algorithm Using Clustering Analysis", in Algorithms, vol. 10, 2017

[18] J. Zhang, H.S.-H Chung, W.-L. Lo, "Clustering-based adaptive crossover and mutation probabilities for genetic algorithms" in *IEEE Trans. Evolut. Comput*, vol. 11, pp 326–335, 2007.

[19] H. Zhang, A. Zhou, S. Song, Q. Zhang, X.-Z. Gao, J. Zhang, "A self-organizing multiobjective evolutionary algorithm" in *IEEE Trans. Evolutionary Computation*, vol. 20, pp 792–806, 2016.

[20] X. Li, S. Song, H. Zhang, "Evolutionary multiobjective optimization with clustering-based self-adaptive mating restriction strategy", in *Soft Computing*, vol. 23, pp 3303, 2019.

[21] E. Zitzler and L. Thiele, "Multiobjective Optimization Using Evolutionary Algorithms – A Comparative Case Study." *In Parallel Problem Solving from Nature*, Vol. 1498, pp 292–301, 1998.

[22] P. Pitiot, L. G. Monge, E. Vareilles, and M. Aldanondo, "Benchmark for Configuration and Planning Optimization Problems: Proposition of a Generic Model." *Proc. of the 18th Inter. Configuration Workshop*, pp 89-96, 2016.