Defence Research and Development Canada | Recherche et développement pour la défense Canada

Integration of Technology Enables for Tactical Picture Compilation
Task 6: Algorithm development/adaptation

Annex 1 - Genetic Algorithms - User's Guide

Mihai Cristian Florea
Thales Canada, Defence and Security

Defence Research and Development Canada – Valcartier

Contract Report
DRDC Valcartier CR 2013-478
December 2013

Canada

Integration of Technology Enables for Tactical Picture Compilation Task 6: Algorithm development/adaptation

Annex 1 - Genetic Algorithms - User's Guide

Mihai Cristian Florea
Thales Canada, Defence and Security

Prepared by:
Thales Canada, Defence and Security
1405, boul. du Parc-Technologique
Québec, QC G1P 4P5

Contractor's document number: 1904C.006-REP-02
PWGSC contract number: W7701-083895/001/QLC

CSA: Anne-Laure Jousselme, (418) 844-4000 x4817

Defence Research and Development Canada – Valcartier

Contract Report
DRDC Valcartier CR 2013-478
December 2013

IMPORTANT INFORMATIVE STATEMENTS

The scientific or technical validity of this Contract Report is entirely the responsibility of the Contractor and the contents do not necessarily have the approval or endorsement of the Department of National Defence of Canada.

Abstract

Within Task 6, our work focused in developing different algorithms (in MATLAB) for the classification problems, as an extension to the DRDC Fusion Toolbox, PRTools, and the FSMOGA Library. Our work has been divided in three sections:

- Investigate how to reduce the high-computation complexity of the DRDC Fusion Toolbox when used in a classification context. This work involved the development of more suitable implementations for the DST combination rules. These rules have also been updated within the DRDC Fusion Toolbox.
- Implement different modelisations, transformations, and combinations (for the DST) using the PRTools mappings concept. The aim of this section was to implement complex PRSC mappings (the PRSC structure has been indicated by the Scientific Authority).
- Extend the MATLAB code for Feature Selection Multi Objective Genetic Algorithm (FSMOGA) provided by the scientific authority in order to use constraints as in the original implementation of the NSGAII algorithm. Extend the code for PRSC optimisation (using the original C code, and using and extending the modelisation proposed by Gabrys and Ruta).

Résumé

Les activités que nous avons réalisées dans le cadre de la Tâche 6 étaient axées sur le développement de divers algorithmes (dans MATLAB) portant sur les problèmes de classification et venant s’ajouter aux outils Fusion Toolbox et PRTools de RDDC, ainsi qu’à la bibliothèque FSMOGA. Notre travail s’articulait autour des trois grands volets ci-dessous.

- Trouver des moyens de simplifier la grande complexité des calculs effectués par l’outil Fusion Toolbox de RDDC lorsqu’il est exploité dans un contexte de classification. Nos efforts portaient ici sur l’élaboration d’implémentations plus appropriées des règles de combinaison de la théorie de Dempster-Shafer (TDS). Ces règles ont également été mises à jour dans l’outil Fusion Toolbox.

- Implémenter diverses modélisations, transformations et combinaisons (en lien avec la TDS) au moyen du concept de mises en correspondance de l’outil PRTools. Les efforts consacrés à ce volet avaient pour but d’implémenter des mises en correspondance complexes dans la configuration des systèmes de reconnaissance des formes (CSRF), la structure de CSRF ayant été précisée par l’autorité scientifique.

- Étendre le code MATLAB de l’algorithme génétique multi-objectif de sélection des caractéristiques (AGMOSC), fourni par l’autorité scientifique, afin de permettre l’application de contraintes, comme c’est le cas avec l’implémentation initiale de l’algorithme NSGA II. Étendre également le code pour permettre l’optimisation de la CSRF (en utilisant le code original en langage C, ainsi qu’une version étendue de la modélisation proposée par Gabrys et Ruta).

This page intentionally left blank.

Table of contents

List of figures

List of tables

This page intentionally left blank.

1 MATLAB Implementation - DRDC Fusion Toolbox and Extensions

1.1 Fusion Toolbox - Bug Correction

Several corrections have been made to the DRDC Fusion toolbox.

- The classes defined within the Fusion Toolbox have to be defined in a class folder : each class in a specific folder.
- The class fbpa had been corrected at lines 680 and 681 (which were commented before) inside the *validatebpaworld* function : when a bpa defined over the powerset was changed from a open world to a closed world, there was an error.
- The class fbpa had been corrected at lines 657 inside the *validatebpaworld* function : when the *obj.world* is *open* and the *world* is also *open*. One command line has been added.
- Within the function *yagr* the subfunction *yagqamatr* had an error which was corrected.

1.2 Fusion Toolbox - DST Combination rules using 3D matrix representation and multiplication

This set of MATLAB functions have been developed in order to compute DST combination rules using precomputed matrix associated to each focal element and to each combination rule. These functions do not use the classes developed in the DRDC Fusion Toolbox, in order to accelerate the computation time.

The principle is simple and will be explained in the following and will be used for different combination rules such as the conjunctive, disjunctive, dempster's, yager's rules (and several more rules).

- First, we pre-compute a 3D matrix associated to each rule and for each size of the frame of discernment.
- Second, using only matrix multiplication operations, we compute the combination of two BPAs, or even of N bpas (using a quasi-associative strategy). The MATLAB function can also be used for batch computations, for example datasets with input arguments represented by a set of sets of BPAs to be combined (represented in a 3D matrix).

This implementation was exploratory, but the use of very large pre-computed 3D matrix.

Even if this implementation uses only matrix computation, which are supposed to be optimized in MATLAB, the use of Smets' implementation of the Fast Transform for the Transferable Belief is faster.

The use of these MATLAB functions is shown in the following example. The creation of the 3D matrix for the conjunctive combination is realized by the following command :

```
conjmat3d(2)

ans(:,:,1) =

     1     1     1     1
     1     0     1     0
     1     1     0     0
     1     0     0     0


ans(:,:,2) =

     0     0     0     0
     0     1     0     1
     0     0     0     0
     0     1     0     0


ans(:,:,3) =

     0     0     0     0
     0     0     0     0
     0     0     1     1
     0     0     1     0


ans(:,:,4) =

     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     1
```

BPAs are represented in columns

```
A =

    0.1000         0         0
```

```
    0.2000    0.2000         0
    0.3000    0.3000    0.5000
    0.4000    0.5000    0.5000


B =

         0         0    0.2000
    0.4000         0    0.5000
         0    0.8000    0.1000
    0.6000    0.2000    0.2000

m = conj3dr(A, B)
m =

    0.2200    0.1600    0.4500
    0.3600    0.0400    0.2500
    0.1800    0.7000    0.2000
    0.2400    0.1000    0.1000
```

Input BPAs can also be stored in a hyper-cube (3D matrix repesentation). And all BPAs in a layer are combined toghether using a quasi-associative rule.

```
C(:,:,1) = A ;
C(:,:,2) = B ;
C(:,:,3) = A ;
m = conjqa3dr(C)

m =
    0.2200    0.1600    0.4500
    0.3600    0.0400    0.2500
    0.1800    0.7000    0.2000
    0.2400    0.1000    0.1000
```

1.3 Fusion Toolbox - Comparison

In this section we will compare the different implementations for the DST toolbox. We consider the following implementations:

- Kennes & Smets' Fast Mobius Transform Tooolbox [1,2]
- Arnaud Martin's Implementation for the DST. This implementation of the combination rules is based mainly on the Smets' FMT transformation toolbox.

– DRDC Fusion Toolbox 3.0 implementation. This implementation is based on classes defined for each data type. The combination rules module is a matrix multiplication version and is based on the use of kroneker matrix. The implementation of the transformation module is based on the pre-computed transformation matrix (Inc, Int, Inc', etc) [3].
– DRDC Fusion Toolbox 4.0 implementation. This implementation is based on classes defined for each data type. The combination rules module is a matrix multiplication version and is based on the use of Smets' FMT toolbox. The implementation of the transformation module is based on Smets' FMT toolbox.
– 3d Matrix Toolbox is a new implementation of the combination rules which is based on pre-computed matrix associated to each combination rule and for each subset of the powerset.

	Smets FMT	Arnaud	DRDC Fusion 3	DRDC Fusion 4
Computation Time	very fast	use FMT		use of FMT
Accepted cardinality	25	25		25

Table 1: *Comparison for the Transformations Module*

Table 2: Combination rules - availability

	Smets FMT	Arnaud	DRDC Fusion 3	DRDC Fusion 4	3d Matrix
conjr	N/A	YES	YES	YES	YES
disjr	N/A	YES	YES	YES	YES
dempr	N/A	YES	YES	YES	YES
yagr	N/A	YES	YES	YES	YES
dpr	N/A	ERROR	YES	YES	YES
dpr2	N/A	NO	YES	YES	YES
pcr5	N/A	YES	YES	YES	YES
pcr6	N/A	YES	YES	YES	YES
rcrsr	N/A	NO	YES	YES	YES
rcrlr	N/A	YES	YES	YES	YES
rcrgr	N/A	NO	YES	YES	YES
inagr	N/A	NO	YES	YES	YES
delmr	N/A	NO	YES	YES	YES
Dubois criteria	N/A	YES NO TEST	NO	NO	NO
Cautious Denoeux min	N/A	YES NO TEST	NO	NO	NO

Continued on next page

Table 2 – continued from previous page

	Smets FMT	Arnaud	DRDC Fusion 3	DRDC Fusion 4	3d Matrix
Cautious Denoeux max	N/A	ERROR execution	NO	NO	NO
Hard Denoeux	N/A	ERROR negative values	NO	NO	NO
Mean BPA	N/A	YES NO TEST	NO	NO	NO

	$\|\Theta\|$								
	2	3	4	5	6	7	8	9	10
conjr	0.0003	0.0003	0.0004	0.0004	0.0005	0.0006	0.0006	0.0008	0.0010
disjr	0.0004	0.0003	0.0004	0.0005	0.0004	0.0006	0.0006	0.0007	0.0009
dempr	0.0003	0.0003	0.0003	0.0005	0.0005	0.0006	0.0007	0.0008	0.0010
yagr	0.0003	0.0003	0.0004	0.0004	0.0005	0.0005	0.0006	0.0008	0.0010
dpr	0.0004	0.0004	0.0004	0.0006	0.0006	0.0006	0.0007	0.0009	0.0011
dpr2	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
pcr5	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
pcr6	0.0026	0.0244	0.2760	5.8969	82.4245				
rcrsr	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
rcrlr	0.0004	0.0004	0.0004	0.0004	0.0006	0.0006	0.0007	0.0009	0.0011
rcrgr	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
inagr	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
delmr	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Dubois criteria	0.0004	0.0003	0.0004	0.0004	0.0005	0.0005	0.0006	0.0007	0.0010
Cautious Denoeux min	0.0006	0.0006	0.0007	0.0008	0.0009	0.0011	0.0012	0.0016	0.0021
Cautious Denoeux max	0.0005	0.0006	0.0008	0.0007	0.0009	0.0011	0.0012	0.0015	0.0020
Hard Denoeux	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Mean BPA	0.0004	0.0003	0.0004	0.0004	0.0004	0.0006	0.0006	0.0006	0.0009

Table 3: *Arnaud Martin's code. Combination of 2 BPAs - 100 Monte-Carlo tests.*

	|Θ|								
	2	3	4	5	6	7	8	9	10
conjr	0.0017	0.0005	0.0005	0.0006	0.0007	0.0009	0.0009	0.0011	0.0014
disjr	0.0005	0.0004	0.0005	0.0006	0.0007	0.0008	0.0009	0.0011	0.0015
dempr	0.0004	0.0004	0.0006	0.0006	0.0007	0.0008	0.0009	0.0011	0.0015
yagr	0.0004	0.0005	0.0006	0.0005	0.0006	0.0008	0.0009	0.0011	0.0015
dpr	0.0004	0.0005	0.0006	0.0007	0.0008	0.0009	0.0010	0.0014	0.0017
dpr2	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
pcr5	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
pcr6	0.0110	4.3958							
rcrsr	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
rcrlr	0.0004	0.0004	0.0006	0.0007	0.0008	0.0009	0.0010	0.0013	0.0017
rcrgr	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
inagr	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
delmr	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Dubois criteria	0.0003	0.0004	0.0005	0.0007	0.0007	0.0008	0.0010	0.0012	0.0017
Cautious Denoeux min	0.0011	0.0009	0.0010	0.0011	0.0014	0.0016	0.0020	0.0024	0.0034
Cautious Denoeux max	0.0008	0.0009	0.0010	0.0012	0.0013	0.0018	0.0020	0.0023	0.0033
Hard Denoeux	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Mean BPA	0.0005	0.0004	0.0005	0.0005	0.0006	0.0008	0.0009	0.0011	0.0014

Table 4: *Arnaud Martin's code. Quasi-Associative Combination of 4 BPAs - 100 Monte-Carlo tests.*

	$\|\Theta\|$								
	2	3	4	5	6	7	8	9	10
conjr	0.0003	0.0003	0.0004	0.0004	0.0005	0.0006	0.0006	0.0008	0.0010
disjr	0.0004	0.0003	0.0004	0.0005	0.0004	0.0006	0.0006	0.0007	0.0009
dempr	0.0003	0.0003	0.0003	0.0005	0.0005	0.0006	0.0007	0.0008	0.0010
yagr	0.0003	0.0003	0.0004	0.0004	0.0005	0.0005	0.0006	0.0008	0.0010
dpr	0.0004	0.0004	0.0004	0.0006	0.0006	0.0006	0.0007	0.0009	0.0011
dpr2	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
pcr5	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
pcr6	0.0026	0.0244	0.2760	5.8969	82.4245				
rcrsr	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
rcrlr	0.0004	0.0004	0.0004	0.0004	0.0006	0.0006	0.0007	0.0009	0.0011
rcrgr	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
inagr	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
delmr	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Dubois criteria	0.0004	0.0003	0.0004	0.0004	0.0005	0.0005	0.0006	0.0007	0.0010
Cautious Denoeux min	0.0006	0.0006	0.0007	0.0008	0.0009	0.0011	0.0012	0.0016	0.0021
Cautious Denoeux max	0.0005	0.0006	0.0008	0.0007	0.0009	0.0011	0.0012	0.0015	0.0020
Hard Denoeux	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Mean BPA	0.0004	0.0003	0.0004	0.0004	0.0004	0.0006	0.0006	0.0006	0.0009

Table 5: *Matrix 3D code. Combination of 2 BPAs - 100 Monte-Carlo tests.*

	|Θ|								
	2	3	4	5	6	7	8	9	10
conjr	0.0006	0.0004	0.0004	0.0004	0.0007	0.0039	0.0265	0.2063	1.6307
disjr	0.0007	0.0003	0.0004	0.0004	0.0005	0.0039	0.0266	0.2072	1.6280
dempr	0.0006	0.0004	0.0003	0.0003	0.0006	0.0038	0.0253	0.1962	1.6316
yagr	0.0004	0.0002	0.0003	0.0004	0.0004	0.0037	0.0253	0.1953	1.6302
dpr	0.0004	0.0005	0.0006	0.0007	0.0008	0.0009	0.0010	0.0014	0.0017
dpr2	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
pcr5	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
pcr6	0.0110	4.3958							
rcrsr	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
rcrlr	0.0004	0.0004	0.0006	0.0007	0.0008	0.0009	0.0010	0.0013	0.0017
rcrgr	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
inagr	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
delmr	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Dubois criteria	0.0003	0.0004	0.0005	0.0007	0.0007	0.0008	0.0010	0.0012	0.0017
Cautious Denoeux min	0.0011	0.0009	0.0010	0.0011	0.0014	0.0016	0.0020	0.0024	0.0034
Cautious Denoeux max	0.0008	0.0009	0.0010	0.0012	0.0013	0.0018	0.0020	0.0023	0.0033
Hard Denoeux	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Mean BPA	0.0005	0.0004	0.0005	0.0005	0.0006	0.0008	0.0009	0.0011	0.0014

Table 6: *Matrix 3D code. Quasi-Associative Combination of 4 BPAs - 100 Monte-Carlo tests.*

2 New MATLAB Mappings

2.1 Simple Uncertainty Modellers Mappings

The mappings presented in this section have been created independetly from the similar functions from the DRDC Fusion Toolbox. These functions are optimized for a use in a PRTools context and do not use the classes from the DRDC Fusion Toolbox. The mappings can have different input arguments (variable) in the same way as the mappings from the PRTools Library. Function *powersetfeatlab.m* is used for all uncertainty modellers mappings in order to compute the labels over the powerset of the classes set.

Table 7: Uncertainty Modellers Mappings - Matlab Files

Filename	Input	Output	Description
sudm.m	N/A OR 1 dataset OR 1 dataset & 1 sudm mapping	1 fixed mapping	Sudano's Classic Uncertainty Modeller [4]
gsmm.m	N/A OR 1 vector (1 or 2 values) OR 1 dataset OR 1 dataset & 1 gsmm mapping OR 1 dataset & 1 vector (1 or 2 values)	1 fixed mapping	Sudano's Generalized Sum Mean Uncertainty Modeller [4]
gpmm.m	N/A OR 1 scalar OR 1 dataset OR 1 dataset & 1 gpmm mapping OR 1 dataset & 1 scalar	1 fixed mapping	Sudano's Generalized Product Mean Uncertainty Modeller [4]
shafm.m			Shafer's Uncertainty Modeller
qlcm.m			Argui *et al.* [5] Uncertainty Modeller

The use of these mappings is shown in the following example

```
W = gsmm ;
W = gsmm(C) ;
W = gsmm(A) ; % is not working - normal
W = gsmm(C, gsmm([3 , 7])) ;
C = (A * Wff) * (A * Wff * Wsc) * Wum ;
```

C is a dataset defined over the powerset of the classes of the dataset A. $(A * Wff)$ is a test dataset. $A * Wff * Wsc$ is the trained mapping.

2.2 DST Combiners Mappings

Different combination rules have been implemented, using the Smets' Fast Transform Toolbox. The implementation is adapted for a PRTools use and do not use the class representation from the DRDC Fusion Toolbox.

Table 8: DST Combiners Mappings - Matlab Files

Filename	Input	Output	Description
dstcombc.m	N/A OR 1 string	1 mapping (combiner)	DST Combiner Mapping. Supported Combinations : 'dempster', 'conjr', 'disjr', 'yagr', 'rcrlr', 'rcrsr', 'rcrgr'. Possible calls : dstcombc('yagr').

The use of these mappings is shown in the following example

```
A = gendatc([50, 50], 12)
Wff = featsel(12, [1 2 6 7 9 11 12]);
Wsc = {ldc, qdc, knnc([], 3)} ;
Wum = gsmm([2 3]) ;
Wcc = dstcombc('dempster') ;
Wcc = dstcombc('rcrgr', '(1-K).^2') ;

X = ((A * Wff) * (A * Wff * Wsc) ) * Wum ;
Y = [X{:}] * Wcc ;
```

2.3 Uncertainty Modellers + DST Combiners Mappings

In this section we consider the combination of classifiers, as presented by Aregui and Denoeux [5]. This combination is realized in 2 steps :

– Uncertainty modellisation of simple classifiers' output
– Combination within the DST

Table 9: Uncertainty Modellers + DST Combiners Mappings - Matlab Files

Filename	Input	Output	Description
qmcdm.m			Rogova's Uncertainty Modeller + Combiner
qmcdcm.m			Rogova's Uncertainty Modeller + Combiner

The use of these mappings is shown in the following example

```
A = gendatc([50, 50], 12)
Wff = featsel(12, [1 2 6 7 9 11 12]);
Wsc =  {ldc, qdc, knnc([], 3)} ;
Wcc = qmcdcm ;

X = (A * Wff) * (A * Wff * Wsc ) ;
Y = [X{:}] * Wcc ;
```

These mappings cannot be used in a list of simple combiners or a list of DST combiners since their behavior is different.

2.4 Decision Mappings

A mapping which is defined by its Uncertainty Modeller, its DST Combination rule and its Decision rule has aslo been proposed.

Table 10: Decision Mappings - Matlab Files

Filename	Input	Output	Description
betpm.m	N/A OR 1 dataset OR 1 dataset & 1 betpm mapping	1 fixed mapping	Maximum Pignistic Probability Decision Mapping
plm.m	N/A OR 1 dataset OR 1 dataset & 1 plm mapping	1 fixed mapping	Maximum Plausibility Decision Mapping

The use of these mappings is shown in the following example

```
A = gendatc([50, 50], 12)
```

```
Wff = featsel(12, [1 2 6 7 9 11 12]);
Wsc =  {ldc, qdc, knnc([], 3)} ;
Wum = gsmm([2 3]) ;
Wcc = dstcombc('dempster') ;
Wcc = dstcombc('rcrgr', '(1-K).^2') ;
Wdec = betpm ;

X = (A * Wff) * (A * Wff * Wsc ) * Wum ;
Y = [X{:}] * Wcc ;
Z = Y * Wdec ;
```

2.5 DST Combiners Mappings - UM / DST Combination / Decision

A mapping which is defined by its Uncertainty Modeller, its DST Combination rule and its Decision rule has aslo been proposed. This mapping has been proposed in order to be used within the PRTools Library, in the same way as the simple combiners from the PRTools.

Table 11: DST Full Combiners Mappings - Matlab Files

Filename	Input	Output	Description
dstc.m	N/A OR 1 cell array	1 mapping (combiner)	DST Combiners Mapping - UM / DST Combination / Decision. Supported UM : 'qlcm', 'shafm', 'gsmm', 'gpmm', 'sudm'. Supported Combinations : 'dempster', 'conjr', 'disjr', 'yagr', 'rcrlr', 'rcrsr', 'rcrgr'. Supported Decisions : 'betpm', 'plm' . Possible calls : dstc('qlc', 'yagr', 'betpm').

The use of these mappings is shown in the following example

```
A = gendatc([50, 50], 12)
Wff = featsel(12, [1 2 6 7 9 11 12]);
Wsc =  {ldc, qdc, knnc([], 3)} ;
Wum = gsmm([2 3]) ;
Wcc = dstc({'qlcm', 'dempster', 'betpm'})
Wcc = {prodc, dstc({'qlcm', 'dempster', 'betpm'})}
```

```
X = (A * Wff) * (A * Wff * Wsc ) ;
Y = [X{:}] * Wcc ;
```

2.6 PRSC Mapping

The PRSC Mapping is used to create a PRTools Mapping according to the PRSC definition. Different situations have been considered.

The input arguments are 5 mappings or list of mappings and the output argument is a PRTools mapping.

Table 12: PRSC Mapping - Matlab Files

Filename	Input	Output	Description
prscm.m	5 mappings OR 3 list of mappings & 2 mappings	1 mapping	PRSC Mapping

The order of the input arguments is the following :
– Wff : feature selection mapping or list of feature selection mappings.
– Wsc : simple classifier mapping or list of simple classifiers mappings.
– Wum : uncertainty modeller mapping or list of uncertainty modeller mappings.
– Wcc : combiner mapping
– Wdec : decision mapping
This mapping can be used by the following calls:

1. prscm(Wff, Wsc, [], [], []) : feature selection only. The first 2 input arguments are mappings.

```
A = gendatc([50, 50], 12)
Wff = featsel(12, [1 2 6 7 9 11 12]);
Wsc = ldc ;
w = prscm(Wff, Wsc, [], [], []) ;

testc(A * (A * w) )
crossval(A, w, 10)
labeld(A * (A * w) )
confmat(A * (A * w))
```

2. prscm(Wff, Wsc, [], Wcc, []) : combination of simple classifiers (using feature selection).
   - Wff can be a mapping, applied to the list of simple classifiers Wsc, and the result should be combined using the simple combiner Wcc.
     ```
     A = gendatc([50, 50], 12)
     Wff = featsel(12, [1 2 6 7 9 11 12]);
     Wsc = {ldc , qdc, knnc([], 3)} ;
     Wcc = prodc ;
     w = prscm(Wff, Wsc, [], Wcc, []) ;

     testc(A * (A * w) )
     crossval(A, w, 10)
     labeld(A * (A * w) )
     confmat(A * (A * w))
     ```
   - Wff can be a list of feature selection mappings, applied to a single simple classifier Wsc, and the result should be combined using the simple combiner Wcc.
     ```
     A = gendatc([50, 50], 12)
     Wff{1} = featsel(12, [1 2 6 7 9 11 12]);
     Wff{2} = featsel(12, [1 4 6 8 11 12]);
     Wff{3} = featsel(12, [1 6 10 11]);
     Wsc = ldc;
     Wcc = prodc ;
     w = prscm(Wff, Wsc, [], Wcc, []) ;

     testc(A * (A * w) )
     crossval(A, w, 10)
     labeld(A * (A * w) )
     confmat(A * (A * w))
     ```
   - Both Wff and Wsc are list of classifiers (with the same number of elements), and the result should be combined using the simple combiner Wcc.
     ```
     A = gendatc([50, 50], 12)
      Wff{1} = featsel(12, [1 2 6 7 9 11 12]);
      Wff{2} = featsel(12, [1 4 6 8 11 12]);
      Wff{3} = featsel(12, [1 6 10 11]);
      Wsc = {ldc , qdc, knnc([], 3)} ;
      Wcc = prodc ;
      w = prscm(Wff, Wsc, [], Wcc, []) ;

      testc(A * (A * w) )
      crossval(A, w, 10)
      labeld(A * (A * w) )
      confmat(A * (A * w))
     ```

- If Wcc is a DST combiner, an error is provided, since an uncertainty modeller is requested.

3. prscm(Wff, Wsc, Wum, Wcc, Wdec) : DST combination of simple classifiers (using feature selection), using uncertainty modellers and decision.
   - Only DST Combiners are allowed by this call.
   - Wff and Wsc can be mappings, and Wum can be a list of mappings. The resulting classifiers are combined through the Wcc. Wdec is applied to the DST combination result.

```
A = gendatc([50, 50], 12)
Wff = featsel(12, [1 2 6 7 9 11 12]);
Wsc = ldc ;
Wum = {gsmm([2 3]) , gpmm, qlcm };
Wcc = dstcombc('dempster') ;
Wdec = betpm ;
w = prscm(Wff, Wsc, Wum, Wcc, Wdec) ;

testc(A * (A * w) )
crossval(A, w, 10)
labeld(A * (A * w) )
confmat(A * (A * w))
```

   - Wff and Wum can be mappings, and Wsc can be a list of mappings. The resulting classifiers are combined through the Wcc. Wdec is applied to the DST combination result.

```
A = gendatc([50, 50], 12)
Wff = featsel(12, [1 2 6 7 9 11 12]);
Wsc = {ldc , qdc, knnc([], 3)} ;
Wum = gsmm([2 3]) ;
Wcc = dstcombc('dempster') ;
Wdec = betpm ;
w = prscm(Wff, Wsc, Wum, Wcc, Wdec) ;

testc(A * (A * w) )
crossval(A, w, 10)
labeld(A * (A * w) )
confmat(A * (A * w))
```

   - Wum and Wsc can be mappings, and Wff can be a list of mappings. The resulting classifiers are combined through the Wcc. Wdec is applied to the DST combination result.

```
A = gendatc([50, 50], 12)
Wff{1} = featsel(12, [1 2 6 7 9 11 12]);
Wff{2} = featsel(12, [1 4 6 8 11 12]);
Wff{3} = featsel(12, [1 6 10 11]);
```

```
Wsc = ldc ;
Wum = gsmm ;
Wcc = dstcombc('dempster') ;
Wdec = betpm ;
w = prscm(Wff, Wsc, Wum, Wcc, Wdec) ;

testc(A * (A * w) )
crossval(A, w, 10)
labeld(A * (A * w) )
confmat(A * (A * w))
```

– Wff can be a mapping, and Wsc and Wum can be lists of mappings (with the same number of elements). The resulting classifiers are combined through the Wcc. Wdec is applied to the DST combination result.

```
A = gendatc([50, 50], 12)
Wff= featsel(12, [1 2 6 7 9 11 12]);
Wsc = {ldc , qdc, knnc([], 3)} ;
Wum = {gsmm([2 3]) , gpmm, qlcm };
Wcc = dstcombc('dempster') ;
Wdec = betpm ;
w = prscm(Wff, Wsc, Wum, Wcc, Wdec) ;

testc(A * (A * w) )
crossval(A, w, 10)
labeld(A * (A * w) )
confmat(A * (A * w))
```

– Wsc can be a mapping, and Wff and Wum can be lists of mappings (with the same number of elements). The resulting classifiers are combined through the Wcc. Wdec is applied to the DST combination result.

```
A = gendatc([50, 50], 12)
Wff{1} = featsel(12, [1 2 6 7 9 11 12]);
Wff{2} = featsel(12, [1 4 6 8 11 12]);
Wff{3} = featsel(12, [1 6 10 11]);
Wsc = ldc ;
Wum = {gsmm([2 3]) , gpmm, qlcm };
Wcc = dstcombc('dempster') ;
Wdec = betpm ;
w = prscm(Wff, Wsc, Wum, Wcc, Wdec) ;

testc(A * (A * w) )
crossval(A, w, 10)
labeld(A * (A * w) )
confmat(A * (A * w))
```

– Wum can be a mapping, and Wff and Wsc can be lists of mappings (with the same number of elements). The resulting classifiers are combined through the Wcc. Wdec is applied to the DST combination result.

```
A = gendatc([50, 50], 12)
Wff{1} = featsel(12, [1 2 6 7 9 11 12]);
Wff{2} = featsel(12, [1 4 6 8 11 12]);
Wff{3} = featsel(12, [1 6 10 11]);
Wsc = {ldc , qdc, knnc([], 3)} ;
Wum = gsmm ;
Wcc = dstcombc('dempster') ;
Wdec = betpm ;
w = prscm(Wff, Wsc, Wum, Wcc, Wdec) ;

testc(A * (A * w) )
crossval(A, w, 10)
labeld(A * (A * w) )
confmat(A * (A * w))
```

– All three input arguments Wff, Wsc and Wum can be lists of mappings (with the same number of elements). The resulting classifiers are combined through the Wcc. Wdec is applied to the DST combination result.

```
A = gendatc([50, 50], 12)
Wff{1} = featsel(12, [1 2 6 7 9 11 12]);
Wff{2} = featsel(12, [1 4 6 8 11 12]);
Wff{3} = featsel(12, [1 6 10 11]);
Wsc = {ldc , qdc, knnc([], 3)} ;
Wum = {gsmm([2 3]) , gpmm, qlcm };
Wcc = dstcombc('dempster') ;
Wdec = betpm ;
w = prscm(Wff, Wsc, Wum, Wcc, Wdec) ;

testc(A * (A * w) )
crossval(A, w, 10)
labeld(A * (A * w) )
confmat(A * (A * w))
```

2.7 Remarks regarding the MATLAB implementation

When using the uncertainty modellers in a sequential mapping, an error occurs because of the *sequential.m* function (which is a PRTools function). Indeed, within the *sequential.m* function, the labels of the features of the final mapping are changed according to the first mapping in the sequence (in some situations). Since the uncertainty modellers functions change the number of features, when used in a sequential mapping, an error occurs because of the missmatch between the number of features of the first mapping and the number of features of the last mapping. In order to solve this problem, the line 132 of the *sequential.m* function :

```
if (isdataset(w)) & ~isempty(featlabels)
```

should be replaced by

```
if (isdataset(w)) & ~isempty(featlabels) & isempty(w.featlab)
```

3 Genetic Algorithms

This section is based on the genetic algorithm NSGA II developed by Deb *et al.* [6].

Our work has started by evaluating two different libraries provided by the Scientific Authority:

- Deb's original code (developed in C) - Revision 1.1.6 (08 July 2011) (for Linux only- 64-bit bug for binary coding fixed): NSGA-II in C with gnuplot (Real + Binary + Constraint Handling)
- FSMOGA MATLAB code developed by DRDC Valcartier.

The Pattern Recognition System Configuration (PRSC) search space (5 dimensions) is done by means of a genetic algorithm (NSGA2). Both implementation of the algorithm have been evaluated for the PRSC context. In this section we will provide the conclusions of our study and the details related to the evolution of the code.

3.1 NSGA2 - C

Next, we present a list of Capabilities and Limitations of this code originally developed by Deb *et al.*.

Capabilities :

- Code in C
- Fast execution if used in C.
- Handling constraints
- Tested and validated by the authors themselves.
- Can be used with binary or real variables (a gene can be composed by both binary and real chromosomes).
- Use of output files to keep track of the tests.

Limitations :

- No immediate integration with the PRTools Toolbox (MATLAB) in order to define new fitness functions for the performance evaluation (objective evaluation).
- The crossover and mutation probability are applied to all binary and real variables (chromozomes). Impossible to set different probabilities for each binary variable for example (without changing the code).
- In a PRSC context, the NSGA2 C code is able to cope with customized fitness functions involving Pattern Recognition capabilities, if all these capabilities are developed in C (trained or untrained classifiers, combiners, uncertainty modellers, decision operators, etc). For the moment, MATLAB code is available for all these functionalities and the integration of the C code with the available MATLAB code was investigated.

Work that has been done :

- Use of the validated results in order to validate other implementations (MATLAB or hybrid - MATLAB and C).

– Use the C code in conjunction with the MATLAB customized fitness function is order to create a hybrid code for the PRSC context.

Remarks on the execution/implementation of the code :

– The configuration of a test is made using an input file or in a console mode.

3.2 NSGA2 - FSMOGA - MATLAB - Provided by the SA

Next, we present a list of Capabilities and Limitations of this code provided by the Scientific Authority.

Capabilities :

– code in MATLAB
– Only the Feature Selection capability within the PRSC is available.
– Possible integration with the PRTools Toolbox (MATLAB) in order to define new fitness functions for the performance evaluation (objective evaluation).

Limitations :

– The constraints are not implemented.
– There is no validation of the results (no comparison with the original code).
– The code was not able to use pre-computation (use with trained classifiers, instead of training the classifiers for each population, for each chromozome, etc).
– The code was not designed to be used with MATLAB's Parallel Toolbox

Work that has been done :

– Find a way to validate the results.
– Explore the implementation of the constraints and the evolution of the code towards the PRSC context.
– Output files have been implemented (similar to the ones provided by the original NSGA2 code)

Possible calls of the original FSMOGA code :

```
run_prscsmoga_test_fsmoga(filename)
run_prscsmoga_test_fsmoga(filename, dataset)
```

Remarks on the execution/implementation of the code :

– The filename in the input arguments is not mandatory. If an empty string is provided (”), no output files will be created. The output files will be found in the folder 'data/output'.
– The test configuration is stored under the 'output/other'.

3.3 NSGA2 - hybrid - MATLAB and C

Usage of the original C code/algorithms proposed by Deb *et al.* [6] was accomplished with as minimal code intrusion as possible.

The main function was made to call a dynamic library function making it available to Matlab through dynamic shared library loading.

The original C code results were then tested against the dynamic shared library version results using the available test cases in the original C code package. Further minimal code intrusion was made to allow automatic testing of these different test cases, without having to re-compile the project. The dynamic shared library was also allowed to select an "alternate" fitness function designed to call Matlab versions which in turn could call this project specific Matlab classification algorithms. The original C code results were then tested against the "alternate" fitness function made to call the same test cases algorithms as the original C code, but translated in Matlab.

Capabilities :
- Code developed in C and interacting with code develped in MATLAB.
- Fast execution for the C components.
- Handling constraints
- In a PRSC context, the NSGA2 hybrid code is able to cope with customized fitness MATLAB functions involving Pattern Recognition capabilities (trained or untrained classifiers, combiners, uncertainty modellers, decision operators, etc).
- Possibility to use untrained classifiers (within the PRSC context) or trained classifiers (pre-computed).

Limitations :
- Limitation of the computation time are imposed by the PRTools capabilities.
- Marshalling between MATLAB/C has a negative impact on the computation time.
- The crossover and mutation probability are applied to all binary and real variables (chromozomes). Impossible to set different probabilities for each binary variable for example (without changing the code).
- The computation time related to the use of precomputed trained classifiers should be investigated.

Work that has been done :
- Interoperability between the C code and the MATLAB code using a dynamic library function made available to Matlab through dynamic shared library loading.
- Validate MATLAB implementations.
- Explore the constraints implementation in the original algorithm in order to code it within MATLAB.

Known issues :
- The hybird MATLAB-C implementation starts a second instance of MATLAB for the computation of the fitness function, which communicates with the C program. We need to have the same pathdef environment for this second instance of MATLAB as for the instance from which we have executed the run_prscsmoga_test_hybridc command.

Possible calls of the hybrid code :

```
run_prscsmoga_test_hybridc(filename)
```

```
run_prscsmoga_test_hybridc(filename, dataset)
```

In order to include more specific fitness functions and constraints handeling, the customization has to be done within the file *nsga2_test_problem.m* in the *nsga2_test_problem_custom* function.

3.4 Pre-computation (trained classifiers) - MATLAB

In order to improve the computation time of the genetic algorithms within a PRSC contex, the pre-computation of the trained classifiers has been explored. Two different alternatives were investigated in order to be implemented with the existing code.

3.4.1 Pre-computation of all possible trained classifiers

One of the investigated alternatives was to precompute all the trained classifiers related to the test. This means that for all classifiers, and for a given training dataset, and for all possible feature selections, the simple untrained classifiers have to be trained. As an example, for a dataset having 20 features, and in a PRSC context using 5 simple classifiers, this means that all 5 classifiers should be trained for all $2^2 0$ possible feature selections.

This alternative has two drawbacks : the precomputing is expensive in terms of computation time and storage of the pre-computed variables it has a great impact on the overall computation time. This alternative has one advantage : the storage and access to the pre-computed trained classifiers is easy and fast.

For each new dataset, we need a precomputing step. In a context of Feature Selection test, the pre-computation of all possible trained classifiers will not reduce the overall test computation time. This could be usefull in more large tests, such as Gabrys and Ruta original or extended schemas. This alternative was implemented with the existing code.

This implementation was made available for the extended FSMOGA implementation and for the extended Gabrys and Ruta algorithms (including the 5D implementation).

The pre-computation was not made available for the hybrid implementation. Within the hybrid implementation, the customized MATLAB fitness function is called with parameters stored in a '.mat' file. Reading the '.mat' file, having the pre-computed classifiers, exploses the computation time of the test (due to the large size of the precomputed classifier list). More modifications to the code should be realized in order to improve the computation time. Due to time/budget constraints, this aspect was not addressed.

3.4.2 Pre-computation of required trained classifiers

Another alternative to the precomputation is to store all trained classifiers, once they are computed the first time. This alternative request more complex modifications to the code and was not yet implemented. There should be a particular interest in how to store the trained classifiers and how to access them, in order to reduce the computation time. This operation should also be synchronized with the use of the Parallel Computing Toolbox of MATLAB (since the update of the list of trained classifiers is realized in a sequential mode and a trade-off between the use of the Parallel Computing Toolbox of MATLAB and the precomputing should be done).

3.4.3 Use of precomputing with the PRSCSMOGA code

By default, the PRSCSMOGA tests are not using precomputation. An optional input argument (string : ’precompute’ or ’no-precompute’) will require the use of the precomputations or not. This option is not available for the original C code, for the hybrid code (MATLAB-C) neither for the original FSMOGA code provided by the SA.

3.5 Parallel computing - MATLAB

The Parallel Computing Toolbox allows the use of up to 12 parallel *workers* in order to improve the computation time, without the need of the MATLAB Distributed Computing Server. The use of the parallel computing toolbox is easy and should be done in three steps:
– Start a MATLABPOOL of a given size with a given profile (local by default)
– use PARFOR instead of FOR command
– close the MATLABPOOL
The MATLAB code within a PARFOR loop should be optimized in order to avoid the communication overload between the different instances of MATLAB.

If more than 12 *workers* are required, the MATLAB Distributed Computing Server should be also used.

3.5.1 Known issues

For MATLAB R2010a or newer, you may experience issues with the new local mpiexec implementation. In order to allow the local scheduler to create and process parallel jobs and matlabpool, you may be required to disable this feature before starting the matlabpool:

```
distcomp.feature( 'LocalUseMpiexec', false )
```

In order to optimize the computation time of the genetic algorithms related to the PRSC context, we need to optimize the number of “*workers*” to be used in the parallel computing process. We also need to find the best use of the parfor loop instead of the for one. In our opinion, the fintness computation should be realiezed using the parallel computing toolbox since each iteration is independent of the others [1].

3.5.2 Use of the parallel computing with the PRSCSMOGA code

By default, the PRSCSMOGA tests are not using parallel computing. An optional input argument (string : ’parallel’ or ’no-parallel’) will require the use of the parallel computing or not. This option is not available for the original C code, for the hybrid code (MATLAB-C) neither for the original FSMOGA code provided by the SA.

3.6 Include constraints in PRSCSMOGA - MATLAB

The constraints have been implemented to be used in different extensions of the genetic algorithm for the PRSC context.

The definition of the constraints is realized in the run_prscsmoga_test file (related to each test type), through the variable CSTR (with the fields .file, .param, .mode and .option).

For the Feature Selection test (fsmoga modified test), a simple function to cope with the constraints has been implemented: *fsmoga_featselconstraints.m*. We can force thus the number of feature for the feature selection genetic algorithm to be less than a given number of features, to be more than a given number of features, or to be within of a specific interval.

For the Gabrys & Ruta original implementation and for its extensions, the simple constraints related to the feature selections are defined in the same *fsmoga_featselconstraints.m* file as for the feature selection simple tests. The only difference here is that the constraints are no more related to only one feature selection buy to the entire layer of the hypercube (all feature selections related to each classifier in the PRSC test).

If other constraints functions need to be implemented they should be coded based on the example of the *fsmoga_featselconstraints.m* function. If more/different input arguments are required for the new functions, then adjustments have to be done in the *prscsmoga.m* and *run_prscsmoga_test.m* functions.

3.7 NSGA2 FSMOGA - extension - MATLAB

The MATLAB implementation of the FSMOGA code have been extened in order to:
– include the constraints into the genetic algorithm
– allow pre-computing and parallel computings

1. The trade-off between the use of the parallel computing and the pre-computing of trained classifiers requires a particular attention to be addressed

– The plot is available for 2 or 3 objectives.
– Output files can be saved if a filename is provided within the input arguments.

```
run_prscsmoga_test_fsmoga_constraints(filename)
run_prscsmoga_test_fsmoga_constraints(filename,dataset)
run_prscsmoga_test_fsmoga_constraints(filename,’precompute’)
run_prscsmoga_test_fsmoga_constraints(filename,...
’no-precompute’)
run_prscsmoga_test_fsmoga_constraints(filename,’parallel’)
run_prscsmoga_test_fsmoga_constraints(filename,...
’no-parallel’)
run_prscsmoga_test_fsmoga_constraints(filename,dataset,...
’precompute’)
run_prscsmoga_test_fsmoga_constraints(filename,dataset,...
’no-precompute’)
run_prscsmoga_test_fsmoga_constraints(filename,dataset,...
’parallel’)
run_prscsmoga_test_fsmoga_constraints(filename,dataset,...
’no-parallel’)
run_prscsmoga_test_fsmoga_constraints(filename,dataset,...
’parallel’, ’precompute’)
run_prscsmoga_test_fsmoga_constraints(filename,dataset,...
’parallel’, ’no-precompute’)
run_prscsmoga_test_fsmoga_constraints(filename,dataset, ...
’no-parallel’, ’precompute’)
run_prscsmoga_test_fsmoga_constraints(filename,dataset, ...
’no-parallel’, ’no-precompute’)
```

3.8 GABRYS and RUTA - original - MATLAB

Gabrys and Ruta [7] proposed to use genetic algorithms to improve the selection across many dimensions of the classifier fusion process including data, features, classifiers and even classifier combiners. They proposed to model the different dimensions as a hyper-cube, which can be seen as a restricted PRSC implementation.

Capabilities :
– Code developed in MATLAB.
– Handling constraints
– From a PRSC point of view, Gabrys and Ruta’s implementation allow the selection of combiners, feature selections, for a given set of simple classifiers.
– Possibility to use untrained classifiers (within the PRSC context) or trained classifiers (pre-computed).

– Possibility to use parallel processing
– Only one objective is coded in order to keep the original implementation
– Output files can be saved if a filename is provided within the input arguments.

Limitations :

– The selection of classifiers is not supported.
– Only simple combiners for PRTools are allowed
– Cannot handle the Uncertainty Modellers, Decision and DST Combiners.
– No plot since only one objective.

Possible calls of the Gabrys & Ruta original code :

```
run_prscsmoga_test_gabrys(filename)
run_prscsmoga_test_gabrys(filename, dataset)
run_prscsmoga_test_gabrys(filename, 'precompute')
run_prscsmoga_test_gabrys(filename, 'no-precompute')
run_prscsmoga_test_gabrys(filename, 'parallel')
run_prscsmoga_test_gabrys(filename, 'no- parallel')
run_prscsmoga_test_gabrys(filename , dataset, 'precompute')
run_prscsmoga_test_gabrys(filename , dataset, ...
'no-precompute')
run_prscsmoga_test_gabrys(filename , dataset, 'parallel')
run_prscsmoga_test_gabrys(filename , dataset, ...
'no-parallel')
run_prscsmoga_test_gabrys(filename , dataset, ...
'parallel', 'precompute')
run_prscsmoga_test_gabrys(filename , dataset, ...
'parallel', 'no-precompute')
run_prscsmoga_test_gabrys(filename , dataset, ...
'no-parallel', 'precompute')
run_prscsmoga_test_gabrys(filename , dataset, ...
'no-parallel', 'no-precompute')
```

3.9 GABRYS and RUTA - extension - MATLAB

Capabilities :

– Code developed in MATLAB.
– Handling constraints
– From a PRSC point of view, Gabrys and Ruta's implementation allow the selection of combiners, feature selections, for a given set of simple classifiers.
– Possibility to use untrained classifiers (within the PRSC context) or trained classifiers (pre-computed).
– Possibility to use parallel processing

– A 3 objectives function has been implemented and more fitness functions can be added.
– The plot is available for 2 or 3 objectives.
– Output files can be saved if a filename is provided within the input arguments.

Limitations :
– The selection of classifiers is not supported.
– Only simple combiners for PRTools are allowed
– Cannot handle the Uncertainty Modellers, Decision and DST Combiners.

Since Gabrys and Ruta [7] was based on a specific fitness function with only one objective, we have proposed here an extension of the model, based on different performance evaluations and fitness functions. Since one population is formed by a given number of multi-dimensional hypercubes (chromosomes) and since in Gabrys and Ruta's model, the hypercube is evaluted not only by the best layer but by all layers, we propose different fitness functions and objectives. Function *gabrys_fitness_modified.m* has been created in order to define 2 or 3 objectives for this test :
– 1st objective can be the mean for all layers (as in Gabrys and Ruta) or cand be defined as the minimum (best) performance among all the layers
– The 2nd Objective is computed as the mean / min / maximum of features associated to the best layer of the hypercube
– The 3rd Objective is computed based on the entire hypercube we first compue the min / max / mean values for each layer second we compute the min / max / mean values for all layers if the option3 has only one element (function), the function used to perform both previous computations is the same.

3.10 GABRYS and RUTA - 5D - MATLAB

Capabilities :
– Code developed in MATLAB.
– Handling constraints
– From a PRSC point of view, Gabrys and Ruta's implementation allow the selection of combiners, feature selections, for a given set of simple classifiers. We have extended this work by performing an optimisation among the uncertaintty modellers, DST combiners and decision functions too.
– Possibility to use untrained classifiers (within the PRSC context) or trained classifiers (pre-computed).
– Possibility to use parallel processing
– classic Gabrys and Ruta fitness and extended fitness customization.
– The plot is available for 2 or 3 objectives.
– Output files can be saved if a filename is provided within the input arguments.

Limitations :
– The selection of classifiers is not supported.
– The selection of Uncertainty Modellers is not supported

3.11 Extensions to the proposed work

This work can be extended in different ways. A few interesting avenues are presented here.

3.11.1 Use of the hybrid implementation

- Use the hybrid implementation in order to improve the selection across many dimensions of the classifier fusion process (PRSC context)
- Include the precomputation of the trained datasets, in an efficient way.

3.11.2 Extend Gabrys and Ruta models

Gabrys and Ruta [7] proposed the modelisation of the classifiers fusion process as a hypercube. This modelisation, on which we have based our extansions to the 5 dimensions model, has some drawbacks since it cannot allow the selection of a random number of simple classifiers or a selection of different uncertainty modellers for different classifiers. The complete Gabrys and Ruta 5d extended model should be extended in order to allow such flexibility.

References

[1] Kennes, R. and Smets, Ph. (1990), Fast algorithms for Dempster-Shafer theory, In B. Bouchon-Meunier, L.A. Zadeh, R.R. Yager, (Ed.), *Uncertainty in Knowledge Bases*, Lecture Notes in Computer Science, pp. 14–23, Springer-Verlag, Berlin.

[2] Kennes, R. and Smets, Ph. (1991), Computational Aspects of the Möbius Transformation, In P.P. Bonissone, L.N. Kanal J.F. Lemmer, M. Henrion, (Ed.), *Uncertainty in Artificial Intelligence 6*, pp. 401–416, Elsevier Science Publishers.

[3] Jousselme, Anne-Laure and Maupin, Patrick (2012), Distances in evidence theory: Comprehensive survey and generalizations, *International Journal of Approx imate Reasoning*, 53, 18–145.

[4] Sudano, J. J. (2002), Inverse Pignistic Transform, In *Proceedings of 5th International Conference on Information Fusion*.

[5] Aregui, A. and Denoeux, T. (2008), Constructing consonant belief functions from sample data using confidence sets of pignistic probabilities, *International Journal of Approximate Reasoning*, 49, 575?594.

[6] Deb, Kalyanmoy, Pratap, Amrit, Agarwal, Sameer, and Meyarivan, T. (2002), A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II, *IEEE Transactions on Evolutionary Computation*, 6(2), 182–195.

[7] Gabrys, Bogdan and Ruta, Dymitr (2006), Genetic algorithms in classifier fusion, *Applied Soft Computing*, 6, 337–347.

DOCUMENT CONTROL DATA

(Security markings for the title, abstract and indexing annotation must be entered when the document is Classified or Designated)

1. ORIGINATOR (The name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Centre sponsoring a contractor's report, or tasking agency, are entered in section 8.) Thales Canada, Defence and Security 1405, boul. du Parc-Technologique Québec, QC G1P 4P5	2a. SECURITY MARKING (Overall security marking of the document including special supplemental markings if applicable.) UNCLASSIFIED 2b. CONTROLLED GOODS (NON-CONTROLLED GOODS) DMC A REVIEW: GCEC APRIL 2011

3. TITLE (The complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S, C or U) in parentheses after the title.)

Integration of Technology Enables for Tactical Picture Compilation Task 6 : Algorithms development/adaptation : Annex 1 - Genetic Algorithms - User's Guide

4. AUTHORS (last name, followed by initials – ranks, titles, etc. not to be used)

Florea, M. C.

5. DATE OF PUBLICATION (Month and year of publication of document.)	6a. NO. OF PAGES (Total containing information, including Annexes, Appendices, etc.)	6b. NO. OF REFS (Total cited in document.)
December 2013	44	7

7. DESCRIPTIVE NOTES (The category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

Contract Report

8. SPONSORING ACTIVITY (The name of the department project office or laboratory sponsoring the research and development – include address.)

Defence Research and Development Canada – Valcartier
2459 Pie-XI Blvd North
Quebec (Quebec)
G3J 1X5 Canada

9a. PROJECT OR GRANT NO. (If appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant.)	9b. CONTRACT NO. (If appropriate, the applicable number under which the document was written.)
	W7701-083895/001/QLC
10a. ORIGINATOR'S DOCUMENT NUMBER (The official document number by which the document is identified by the originating activity. This number must be unique to this document.)	**10b. OTHER DOCUMENT NO(s).** (Any other numbers which may be assigned this document either by the originator or by the sponsor.)
1904C.006-REP-02	DRDC Valcartier CR 2013-478

11. DOCUMENT AVAILABILITY (Any limitations on further dissemination of the document, other than those imposed by security classification.)

Unlimited

12. DOCUMENT ANNOUNCEMENT (Any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in (11) is possible, a wider announcement audience may be selected.))

Unlimited

13. ABSTRACT (A brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual.)

Within Task 6, our work focused in developing different algorithms (in MATLAB) for the classification problems, as an extension to the DRDC Fusion Toolbox, PRTools, and the FSMOGA Library. Our work has been divided in three sections:

- Investigate how to reduce the high-computation complexity of the DRDC Fusion Toolbox when used in a classification context. This work involved the development of more suitable implementations for the DST combination rules. These rules have also been updated within the DRDC Fusion Toolbox.
- Implement different modelisations, transformations, and combinations (for the DST) using the PRTools mappings concept. The aim of this section was to implement complex PRSC mappings (the PRSC structure has been indicated by the Scientific Authority).
- Extend the MATLAB code for Feature Selection Multi Objective Genetic Algorithm (FSMOGA) provided by the scientific authority in order to use constraints as in the original implementation of the NSGAII algorithm. Extend the code for PRSC optimisation (using the original C code, and using and extending the modelisation proposed by Gabrys and Ruta).

Les activités que nous avons réalisées dans le cadre de la Tâche 6 étaient axées sur le développement de divers algorithmes (dans MATLAB) portant sur les problèmes de classification et venant s’ajouter aux outils Fusion Toolbox et PRTools de RDDC, ainsi qu’à la bibliothèque FSMOGA. Notre travail s’articulait autour des trois grands volets ci-dessous.

- Trouver des moyens de simplifier la grande complexité des calculs effectués par l’outil Fusion Toolbox de RDDC lorsqu’il est exploité dans un contexte de classification. Nos efforts portaient ici sur l’élaboration d’implémentations plus appropriées des règles de combinaison de la théorie de Dempster-Shafer (TDS). Ces règles ont également été mises à jour dans l’outil Fusion Toolbox.
- Implémenter diverses modélisations, transformations et combinaisons (en lien avec la TDS) au moyen du concept de mises en correspondance de l’outil PRTools. Les efforts consacrés à ce volet avaient pour but d’implémenter des mises en correspondance complexes dans la configuration des systèmes de reconnaissance des formes (CSRF), la structure de CSRF ayant été précisée par l’autorité scientifique.
- Étendre le code MATLAB de l’algorithme génétique multi-objectif de sélection des caractéristiques (AGMOSC), fourni par l’autorité scientifique, afin de permettre l’application de contraintes, comme c’est le cas avec l’implémentation initiale de l’algorithme NSGA II. Étendre également le code pour permettre l’optimisation de la CSRF (en utilisant le code original en langage C, ainsi qu’une version étendue de la modélisation proposée par Gabrys et Ruta).

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (Technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus, e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

DRDC Fusion Toolbox; PRTools; FSMOGA Library; DST combination rules; Feature Selection Multi Objective Genetic Algorithm (FSMOGA); NSGAII algorithm; DST Combiners mappings

Defence R&D Canada

Canada's Leader in Defence
and National Security
Science and Technology

R & D pour la défense Canada

Chef de file au Canada en matière
de science et de technologie pour
la défense et la sécurité nationale

DEFENCE DÉFENSE

www.drdc-rddc.gc.ca