NOUVEAU RACCORD

Réunion faite le 13/09/2013, en présence de :
Marianne Brouty, Catherine Brunet, Cécile Loup, Anaïs Oberto, François Ochsenbein, Emmanuelle Perret, Bernd Vollmer

Etat de fait : *(9 septembre)*

Calcul de vraisemblance entre 2 données identiques (1 candidat Simbad et 1 donnée de la Table)

Score = $1.5-0.5\times\left(\frac{|valeurSimbad-valeurTable|}{\sqrt{sSimbad^2+sTable^2}}\right)$

et bornés à [-1 ; 1]

Pour la comparaison des données → remise arrondi dans un même référentiel (Mag AB → Vega , Vitesse → Redshift)
Dans le cas où une erreur est manquante → estimation

Cas particuliers :

- Coordonnées : estimation de l'erreur manquante en effectuant une projection des coordonnées de l'un vers l'autre grâce à une rotation avec l'ellipse d'erreur présente
- Magnitudes :utilisation du caractère de variabilité si présent
- Type d'objet : Compatibilité ou non (dans la même branche ou plus bas)
- Vitesse : estimation de l'erreur manquante selon la valeur :
  - redshift : s=0.01
  - <250 : s=50
  - <5000 : s=0.2*valeur
  - sinon s=1000
- Dimensions : s=0.1' pour petit et grande axe (a25 et b25) et pour l'angle :
  Compare the PA values. Use the sin(PA) with
  sigma(|sin(PA)|) = 10**(-R(D+0.75))
  inclinaison = 1.5 ---> pas encore implémenté
- Parallaxe : estimation de l'erreur à 7mas
- Mouvement propre et coordonnées : applique un minimum pour l'erreur (1 pour PM, en fonction de la précision pour les coo :

```
360.*3600.,     /* k=0: Unknown */
3600.,          /* k=1: degre   */
0.2*3600.,      /* k=2: .1degre */
150.,           /* k=3: '       */
18.,            /* k=4: .1'     */
3.,             /* k=5: "       */
1.0,            /* k=6: 0.1"    */
0.6,            /* k=7: 0.01"   */
0.3,            /* k=8: mas     */
0.3,            /* k=9: 0.1mas  */
```

- Identificateurs : +1 par id trouvé

Possibilité d'appliquer un facteur de tolérance → revient à augmenter ou diminuer la vraisemblance donc influe aussi sur le score pour la donnée.

Jusqu'à maintenant le score est une somme de toutes les vraissemblances pour un objet et l'utilisateur donnait **une** borne min et max pour qu'il prenne la décision de raccordement ou non.
Nous avons maintenant la possibilité de donner des règles sur le score par type de donnée.

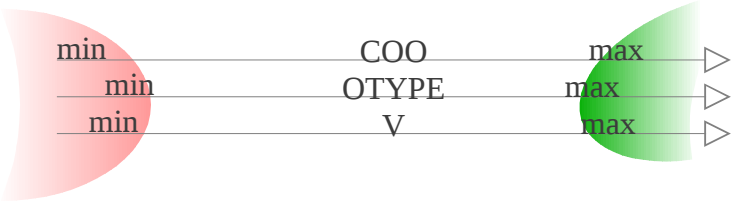
Comportement de la nouvelle version :

1ère étape

Comme avant, le programme commence par interroger par identificateur principal (%I.0) et les autres identificateurs si donnés (%I), si rien n'est trouvé, il cherche par coordonnées. Ensuite, il compare chaque objet trouvé par rapport à l'objet entré et évalue sa vraisemblance (score) à être un bon candidat à cross-identification.

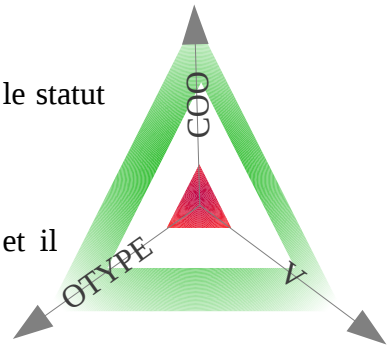
2ème étape

L'utilisateur donne ses limites min max pour chaque score (pour l'instant COO, OTYPE et V) :
Si scoreX < minX alors MAUVAIS score (trop bas)
Si scoreX >= maxX alors BON score (assez haut)
Par défaut les bornes sont à -1 et 1



Algorithme de décision du statut du candidat

C'est le résultat de la combinaison des différents scores qui va renseigner sur le statut d'un candidat (BON ou MAUVAIS).
Si **tous** les scores sont hauts alors le candidat est considéré comme BON.
Si **tous** les scores sont bas alors le candidat est considéré comme MAUVAIS.
Dans tous les cas intermédiaires, le programme ne prend pas de décision et il affiche les candidats.

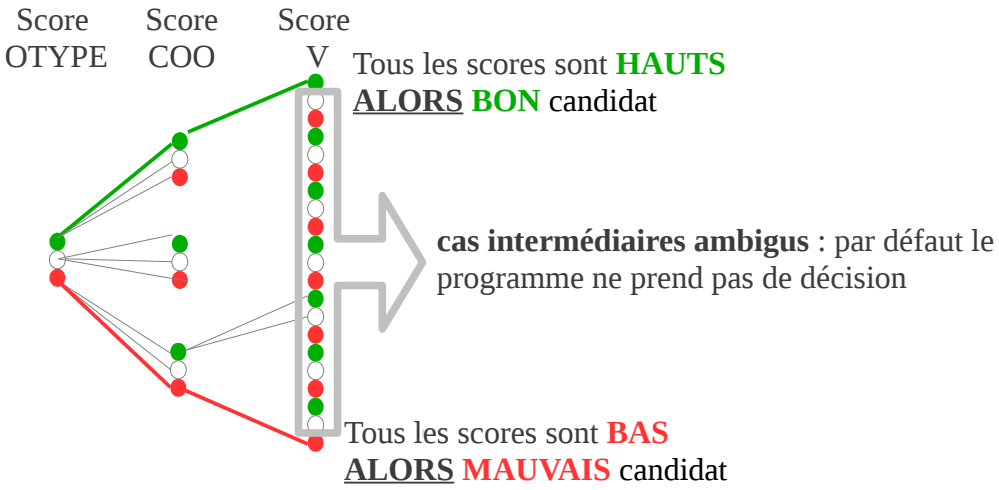


ex :

```
.SCO_COO -2,2
.SCO_OT -1,1
.SCO_V -1,1
```

SI le score des coordonnées est < -2 **ET** le score du type d'objet est < -1 **ET** le score de la vitesse est < -1 **ALORS** MAUVAIS candidat
SI le score des coordonnées est >=2 **ET** le score du type d'objet est >= 1 **ET** le score de la vitesse est >= 1 **ALORS** BON candidat

On peut représenter le comportement sur le schéma suivant :



Le travail de la documentaliste est d'étudier le nombre de cas où le programme ne décide rien. Cela se fait à deux endroits :

1. en modifiant les valeurs min et max de chaque score afin de réduire la zone d'incertitude de chaque score (zone blanche dans le schéma « pyramidal ») ; dans l'idéal jusqu'à obtenir min=max ;
2. en indiquant au programme les comportements à adopter dans tous les cas ambigus (par ex : 2 scores bons, et 1 mauvais)

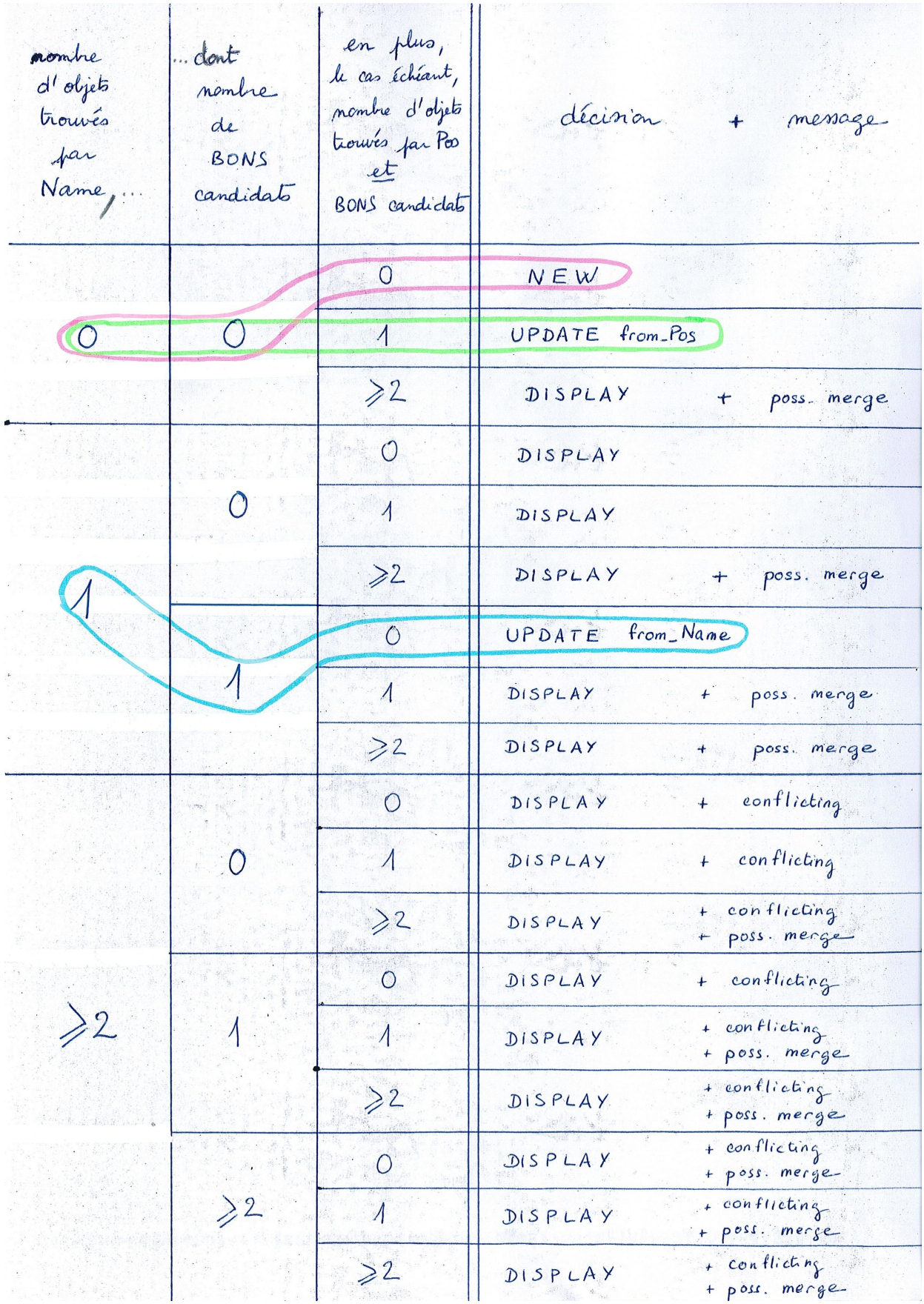
Génération des commandes de mises à jour

Ensuite, selon la découverte de bons candidats, le programme va conclure pour donner les commandes de mises à jour :

- UPDATE de l'objet Simbad en cas de cross-identification confirmée par de bons scores
- Nouvel objet créé avec les données de la table d'entrée

Les cas particuliers à noter sont :

- Si aucun objet n'a été trouvé (ni par nom, ni par coordonnées) → Nouvel objet
- Si aucun objet n'a été trouvé par nom mais 1 seul bon candidat trouvé par coordonnées → UPDATE From Pos
- Si 1 seul objet a été trouvé par nom et qu'il est un bon candidat, et qu'aucun bon candidat n'est trouvé par coordonnées → UPDATE From Name
- Dans les autres cas on a les messages d'alertes :
  - « possible merge » : dès qu'on a 2 BONS candidats quelle que soit la façon dont ils ont été trouvés
  - « conflicting » : dès que la recherche par Name donne 2 objets ou plus (qu'ils soient bons ou mauvais candidats)
- On veut pouvoir outrepasser tous les cas et rester en « DISPLAY » dès qu'un identificateur donné est mal écrit (acronyme ou format non reconnu par Simbad)



Statistiques

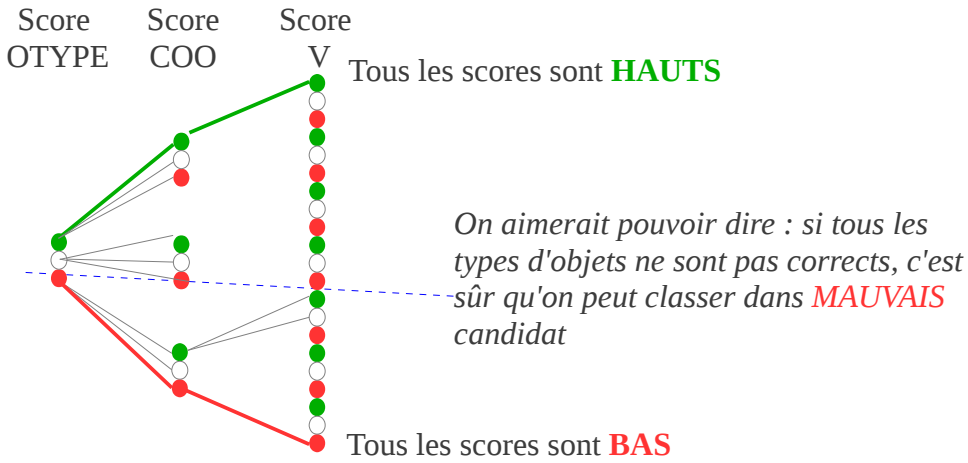
A la sortie de raccord, un fichier de statistiques est alors généré (qui vient remplacer l'actuel script

« statrac ») permettant de savoir :

- le nombre de décisions de nouveaux objets / raccordement,
- le nombre d'identificateurs mal reconnus,
- la liste des types des objets Simbad mis à jour avec leur fréquence selon le type d'objet donné
- la liste des candidats rejetés triés par distance
- …

Points à améliorer :

<u>Outils pour résoudre les cas ambigus</u>



Idée pour diminuer le nombre de cas ambigus : en écartant des cas bien définis, par exemple : donner une condition où l'on sait que ce ne sont pas de bons candidats au raccordement en plaçant un « ! » après la valeur du min pour chaque donnée nécessairement bonne. Ce qui équivaut à dire qu'un score en dessous du « ! » est éliminatoire pour être classé bon. → non, trop compliqué
Autre suggestion : écrire explicitement des règles logiques qui conduisent au choix du statut du candidat.

<u>Que faire quand un score ne peut pas être calculé ?</u>

Quand une donnée est absente (dans Simbad ou le parfile entré), la comparaison devient impossible. Que faut-il alors prendre comme décision ?
Suggestions :

Par défaut → Aucune prise de décision + option pour donner une valeur fictive de score

<u>Définir deux comportements différents selon si trouvé par ID / COO</u>

Il faut un moyen pour mieux utiliser le score des identificateurs (+1 par ID trouvé ou plus/moins si option selon une liste d'acronymes donnés par l'utilisateur)

Suggestion d'écriture d'un algorithme de décision :

ex :

```
.SCO_ID >=1
..SCO_COO -5,0

.SCO_COO -2,2
.SCO_V -1,1
[...]
```

Si le score des ID >=1 (souvent équivalent à « trouvé par identificateur »)
ALORS les bornes pour les coordonnées sont :-5 et 0 (présence des « **..** » marque le fait qu'on est sous la condition)
Sinon (on retrouve une ligne normale) les bornes des coordonnées seront -2 et 2.

Remarques :

- Faire systématiquement un « DISPLAY » si au moins un identificateur donné est mal écrit (et prévoir l'option pour laisser l'identificateur de côté) <=> inverse de l'ancien raccord
- On pourrait tenir compte de la qualité si <C pour estimer les erreurs des coo et vitesses (algo à définir plus tard quand ce sera complet)
- Faire une librairie externe avec ces conversions et estimations.
- Comparaison des types d'objets : modifier la méthode pour utiliser un tableau de compatibilité (OUI = 1 / NON = -1 / ? = 0) qui sera à faire et maintenir par les astronomes.
- Lorsqu'il faut modifier l'objet Simbad, la méthode pour choisir le type d'objet principal final est à revoir (la même que pour les « merge » dans la mise à jour)
- Nouveau message d'alerte en sortie : « possible merge » si plusieurs bons candidats (quelle que soit l'origine) et ne pas prendre de décision (faudra t'il une option pour outrepasser ce comportement?)
- Option pour forcer l'erreur max venant de Simbad (pour palier aux mauvaises données anciennes dans Simbad) pour chaque donnée
- Lister systématiquement TOUS les objets, même s'il y en a plus que 2 dans les cas de « conflicting » (voir tableau).
- Prévoir dans l'interface de sélectionner des « groupes » (« ceux qui passent » ou les « conflicting » ….) pour uniquement afficher ceux là
- Ne pas faire intervenir autant de scores qu'il y a de magnitudes à comparer, mais plutôt 1 score d'une seule magnitude (choisie dans une liste de priorités).
- Suggestion d'avoir un ensemble de règles logiques à réutiliser nommé avec des alias : « naines brunes », « objets Radio » etc...
- Pourquoi ne pas avoir une seule borne pour les scores (équivalent à min=max) ? → a priori non, mais à tester sur des cas concrets, pourrait simplifier
- Faire de nouveaux tests mi octobre.