Zeitschrift: Bulletin de la Société Vaudoise des Sciences Naturelles
Herausgeber: Société Vaudoise des Sciences Naturelles
Band: 80 (1990-1991)
Heft: 3

Artikel: BioGraph : un nouveau programme de construction des corrélations biochronologiques basées sur les associations unitaires

Autor: Savary, Jean / Guex, Jean

DOI: https://doi.org/10.5169/seals-279566

BULL. SOC. VAUD. SC. NAT. 80.3, 1991: 317-340

BULLETIN N° 313 des Laboratoires de Géologie, Minéralogie, Géophysique
et du Musée géologique de l'Université de Lausanne

BioGraph: un nouveau programme de construction des corrélations biochronologiques basées sur les associations unitaires[1]

PAR

JEAN SAVARY[2] et JEAN GUEX[3]

Résumé.-SAVARY J., GUEX J., 1991. BIOGRAPH: un nouveau programme de construction des corrélations biochronologiques basées sur les associations unitaires. *Bull. Soc. vaud. Sc. nat. 80.3*: 317-340.
Un nouveau programme de construction des Associations Unitaires (A.U.) et des corrélations biochronologiques est décrit ici. Ce programme, nommé BIOGRAPH, est basé sur de récents développements de la méthode des A.U. Il permet l'établissement rapide d'une échelle biochronologique discrète sur n'importe quel micro-ordinateur IBM PC ou compatible.

mots clefs: associations unitaires, biochronologie, stratigraphie.

Abstract.-SAVARY J., GUEX J., 1991. BIOGRAPH: a new program for constructing biochronologic correlations based on the unitary associations. *Bull. Soc. vaud. Sc. nat. 80.3*: 317-340.
A new program for constructing Unitary Associations (U.A.) and biochronologic correlations is described here. This program, named BIOGRAPH, is based on recent developments on the U.A. method. It is designed to establish discrete biochronologic relative time scales calculated on any IBM PC or compatible microcomputer in very short time.

keywords: unitary associations, biochronology, stratigraphy.

---

[1]Travail publié dans le cadre du projet 2.685.080 du Fonds National Suisse pour la Recherche Scientifique.
[2]La Crettaz, CH-1965 Savièse.
[3]Institut de Géologie et de Paléontologie, Université de Lausanne, BFSH-2, CH-1015 Lausanne.

ISSN 0037-9603
CODEN: BSVAA6

1. Introduction

Une échelle biochronologique basée sur le contenu paléontologique des roches sédimentaires est une représentation synthétique de toutes les relations stratigraphiques observées (dans un état de connaissance donné) entre les espèces fossiles. Une échelle construite à partir des associations unitaires (détails in Guex 1987) est qualifiée de «discrète» car les intervalles qui constituent ses subdivisions ne sont pas contigus.

Pour être utilisable en pratique, une telle échelle doit satisfaire deux conditions:

-elle doit rendre un compte exact de toutes les coexistences observées entre les espèces (à l'exclusion des remaniements et des condensations);

-elle ne doit contredire aucune superposition stratigraphique observée (i.e. les superpositions stratigraphiques conflictuelles y apparaissent commes des coexistences virtuelles entre les espèces).

Construire une échelle qui satisfasse ces critères est souvent difficile car les relations stratigraphiques entre les espèces sont généralement affectées par une multitude de contradictions apparentes.

L'analyse de ces contradictions ainsi que la façon de les interpréter a été abordée à plusieurs reprises par l'un de nous (Guex 1987, 1988). Les démarches algorithmiques suivies dans ces travaux ont été formulées en termes de théorie des graphes. La terminologie technique empruntée à cette théorie ainsi que les notations utilisées ci-dessous sont conformes à celle de Roberts (1976) et nous n'y reviendrons pas ici.

Les solutions qui ont été proposées pour résoudre ce problème ont en commun l'analyse individuelle de chaque contradiction pour en déduire les coexistences virtuelles (coexistences chronologiques déduites) entre les espèces. L'introduction de ces coexistences virtuelles dans le graphe représentatif des relations stratigraphiques entre les espèces (graphe biostratigraphique G*) permet de transformer ce graphe en un graphe d'intervalles.

Deux procédures d'optimisation ont été étudiées en détail dans les travaux sus-mentionnés. La première, due à Eric Davaud (in Guex et Davaud 1982), cherche à préserver le plus grand nombre d'informations sur les superpositions stratigraphiques observées localement entre les espèces (minimisation du nombre d'arcs de G* remplacés par des arêtes virtuelles). La deuxième (Guex 1987, 1988) utilise uniquement la reproductibilité des arcs (i.e.: le nombre de profils stratigraphiques dans lesquels une superposition entre deux espèces est observée) comme critère d'optimisation.

Nous avons développé un programme, nommé BioGraph, automatisant la méthode et qui évite les difficultés inhérentes aux approches combinatoires discutées dans les travaux sus-mentionnés. Ce programme est en partie basé sur une méthode nouvelle décrite par Guex (1988). Il fonctionne sur les ordinateurs personnels compatibles IBM. Il peut être obtenu auprès des auteurs.

La présente note a pour but de décrire l'approche méthodologique et les aspects informatiques de ce programme en suivant l'algorithme représenté à la figure 1.

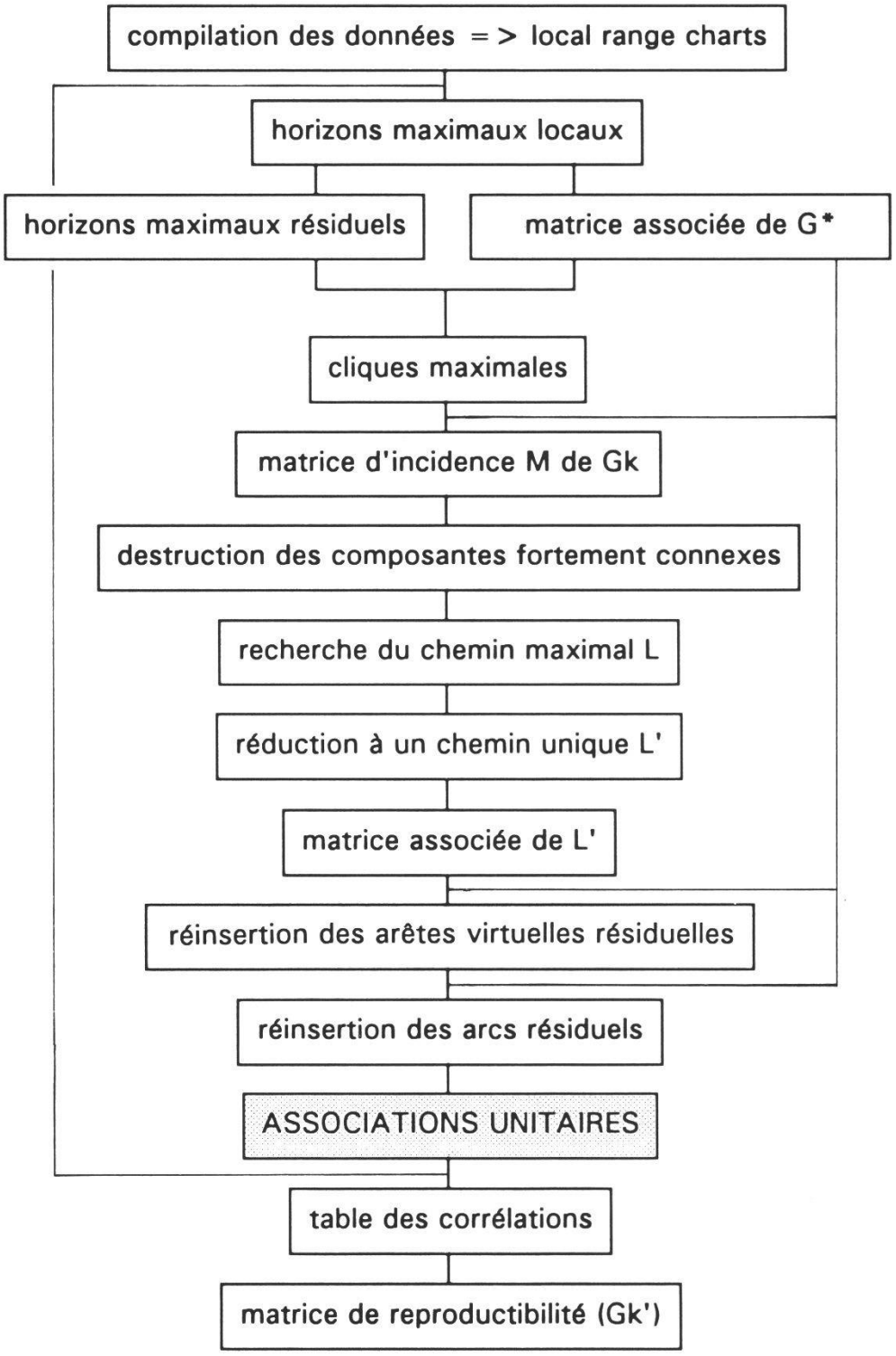

Figure 1.-Algorithme de BioGraph.

2. Description méthodologique

2.1. *Conventions et définitions*

-Une espèce fossile montrant une distribution verticale discontinue dans un profil stratigraphique, est considérée comme virtuellement présente dans tous les horizons qui sont encadrés par son apparition et sa disparition locales.

-Dans un profil stratigraphique donné, seuls sont retenus les niveaux qui enregistrent les intersections maximales (par rapport à la relation d'inclusion) d'intervalles d'existence locaux des espèces. Ces niveaux sont appelés des horizons maximaux. Les horizons maximaux qui sont strictement distincts les uns des autres (par rapport à la relation d'inclusion) dans un ensemble de profils stratigraphiques sont appelés des horizons maximaux résiduels (en bref: horizons résiduels).

2.2. *Méthode*

2.2.1. *Construction des cliques maximales de G**

Une fois les horizons maximaux résiduels et la matrice associée à G* construits, les cliques maximales de G* sont fabriquées à partir des horizons résiduels de la façon suivante.

Soit k_m l'horizon résiduel de plus grande cardinalité.

Soit $\Gamma(x_i)$ l'ensemble des sommets voisins de x_i dans G*.

-Si $k_m - \Gamma(x_i) = \varnothing$, x_i est ajouté à k_m (x_i est compatible avec tous les éléments de k_m). k_m est alors noté $k_m^* = \{k_m \cup x_i\}$.

-Le prochain élément x_j ajouté à k_m^* est celui pour lequel $k_m^* - \Gamma(x_i) = \varnothing$ (km^* devient alors égal à $\{km \cup x_i \cup xj\}$). k_m^* augmenté est ainsi successivement comparé à tous les sommets de G*.

-Au terme de cette opération, tous les horizons résiduels qui sont contenus dans k_m^* sont éliminés puis la procédure se poursuit pour le k_m suivant etc.

Les k_m^* résultants sont maintenant des cliques distinctes de G*.

Cette approche ne produit pas une liste exhaustive des cliques maximales mais évite une croissance exponentielle de leur nombre.

2.2.2. *Relations stratigraphiques entre les cliques maximales*

La relation stratigraphique entre deux cliques maximales, k_i et k_j, peut être déduite des relations stratigraphiques observées entre les espèces appartenant respectivement à k_i et à k_j. Il existe trois types de relations entre ces cliques:

-La clique k_i se situe au-dessus (ou au-dessous) de k_j s'il existe au moins une espèce de k_i qui est située au-dessus (ou au-dessous) d'une espèce de k_j (superposition univoque).

-La relation entre k_i et k_j est indéterminée si les espèces de k_i ne sont pas compatibles avec celles de k_j et si leurs relations superpositionnelles sont indéterminées.

-La relation entre k_i et k_j est conflictuelle si certaines espèces de k_i sont situées au-dessus de certaines espèces de k_j et inversement.

2.2.3. *Recherche et éliminations des contradictions*

Pour transformer un graphe biostratigraphique en un graphe d'intervalles en ordonnant ses cliques maximales à partir des observations biostratigraphiques brutes, il est indispensable d'éliminer les superpositions contradictoires entre les espèces. Ceci peut être fait de deux façons. La première consiste à remplacer certaines superpositions par des coexistences virtuelles (voir plus haut). La seconde ignore certaines superpositions contradictoires et peu reproductibles.

La méthode sur laquelle notre nouveau programme est fondée part du principe suivant: étant données deux superpositions conflictuelles, nous sommes forcés (en l'absence d'arguments non biochronologiques) d'admettre que la superposition reproductible (i.e. observée sur une grande étendue géographique) a plus de «valeur» chronologique que celle qui ne l'est pas. Autrement dit, si deux cliques distinctes de G* contiennent respectivement des espèces qui les situent à la fois au-dessus et au-dessous l'une de l'autre, dans chaque cas sera considérée comme «vraie» la superposition reproductible et comme «fausse» (i.e.: induite par une documentation insuffisante ou par un remaniement) celle qui l'est moins.

En pratique le procédé sera le suivant. Pour chaque couple k_i, k_j de cliques maximales de G* montrant une relation stratigraphique contradictoire, deux ensembles d'arcs sont définis, A et B, où A est l'ensemble des arcs qui relient des éléments de k_i à ceux de k_j (dans le sens $k_i \rightarrow k_j$) et où B est l'ensemble des arcs (d'orientation inverse) qui relient des éléments de k_j à ceux de k_i. Chaque ensemble A et B possède une valeur V(A) (respectivement V(B)) égale à la somme des reproductibilités individuelles des arcs appartenant à A (respectivement B) ajoutée au nombre d'arcs de A (respectivement B).

-Si V(A) > V(B) la clique k_j est «située au-dessus» de k_i (en notant bien qu'il s'agit là d'un abus de langage et que la superposition est conflictuelle).

-Si V(A) = V(B) la relation stratigraphique entre k_i et k_j est indéterminée.

La détection de ces relations conflictuelles met en évidence l'existence de «configurations interdites» dans le graphe biostratigraphique, c'est-à-dire la présence de circuits et de cycles d'ordre 3 ou 4 (fig. 2).

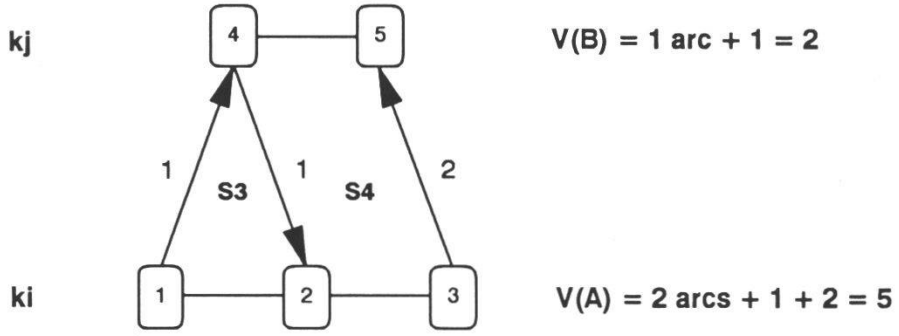

Figure 2.-Exemple de relation conflictuelle entre deux cliques maximales.

2.2.4. *Matrice d'incidence M et graphe Gk*

Une fois établies, les relations stratigraphiques entre les cliques sont compilées dans une matrice d'incidence M, «clique - clique».

Les relations représentées dans M sont symbolisées ainsi:

$-k_i \Leftarrow k_j$ = k_i au-dessus de k_j sans contradiction;

$-k_i \Rightarrow k_j$ = k_i au-dessous de k_j sans contradiction;

$-k_i \leftarrow k_j$ = k_i au-dessus de k_j avec contradiction ($V(A) > V(B)$);

$-k_i \rightarrow k_j$ = k_i au-dessous de k_j avec contradiction ($V(A) < V(B)$);

$-k_i \rightleftharpoons k_j$ = indétermination ($V(A) = V(B)$) ou indétermination pure.

Partant de cette matrice, on construit ensuite le graphe orienté Gk qui lui est associé.

2.2.5. *Les composantes fortement connexes de Gk*

2.2.5.1. *Détection et destruction*

Les données biostratigraphiques mal documentées (polluées par des remaniements non détectés ou par des identifications taxonomiques fausses) sont souvent à l'origine de la présence de composantes fortement connexes (cycles) dans Gk. Ces composantes doivent naturellement être éliminées.

L'algorithme utilisé pour les détecter est voisin de celui qui est décrit par CARRE (1979, p. 47): nous n'y reviendrons pas ici.

Dans notre programme, la destruction des composantes connexes est effectuée selon la règle suivante:

Considérons, dans une composante donnée de Gk, l'ensemble des couples (k_i, k_j) de cliques dont la relation stratigraphique déduite est de type conflictuel (i.e.: $\rightarrow$ ou $\leftarrow$ dans la matrice M). Notons A l'ensemble des arcs qui relient k_i à k_j et notons B l'ensemble des arcs d'orientation inverse. Chaque ensemble A et B possède une valeur V(A) (respectivement V(B)) égale à la somme des reproductibilités individuelles des arcs appartenant à A (respectivement V(B)) (voir plus haut). Chaque arc $k_i \rightarrow k_j$ est alors valué ainsi:

$$C_{ij} = \min(V(A),V(B)) / \max(V(A),V(B))$$

L'arc $k_i \rightarrow k_j$ pour lequel la valeur C_{ij} est la plus élevée est détruit (la relation k_i, k_j est alors considérée comme indéterminée). Si cette indétermination ne suffit pas à détruire la composante connexe, le couple (k_i, k_j) suivant subit le même sort, et ainsi de suite.

Lorsque le problème traité est mal documenté et présente peu de contrôles stratigraphiques, les valeurs de reproductibilité sont faibles et les paramètres V(A) et V(B) sont peu contrastés. En conséquence, les composantes fortement connexes prennent vite de l'ampleur et il existe non pas un seul arc mais un groupe d'arcs possédant la même valeur C_{ij} élevée. Dans ce cas, notre programme brise tout le groupe d'arcs.

2.2.5.2. *Remarque technique*

Il est nécessaire d'insister ici sur le fait que, dans le cas de données particulièrement médiocres, la destruction des composantes connexes se fait de façon quasiment arbitraire et dépend alors de l'ordre selon lequel Gk est parcouru. De tels cas peuvent être détectés en permutant l'ordre des sections stratigraphiques dans le fichier des données (ce qui a pour effet de modifier l'indexation des horizons maximaux et par conséquent celle des sommets de Gk). Si les solutions obtenues diffèrent d'une fois à l'autre, il est alors nécessaire de générer artificiellement des coexistences virtuelles en groupant les associations unitaires dont les éléments caractéristiques ont été permutés: les positions différentes qui leur sont assignées dans les différentes solutions testées résultent du fait que leurs relations chronologiques réelles sont totalement indéterminables. Notons finalement que, dans un travail plus ancien (GUEX et DAVAUD 1984), nous avions renoncé à traiter les problèmes biochronologiques dont les données généraient des circuits dans Gk. Si nous avons «obligé» le présent programme à traiter n'importe quel type de données, si mauvaises soient-elles, c'est parce que de telles structures sont en fait moins rares que nous le pensions alors. L'utilisateur dispose donc ici d'un outil qui lui permet d'analyser en détail les points les plus faibles de sa documentation biostratigraphique et de recalculer ses corrélations autant de fois que nécessaire.

2.2.6. *Chemin maximal de Gk*

Lorsque le graphe obtenu au terme de ces opérations est constitué par plusieurs chemins parallèles, il est réduit à un chemin maximal unique en utilisant une technique voisine du *best fit* décrite en détail par GUEX (1987, p. 83-84). Dans le programme BIOGRAPH, cette technique a été complétée par l'adjonction d'une contrainte supplémentaire: un sommet qui n'appartient pas au plus long chemin de Gk (L) ne peut être réuni qu'avec un successeur de son prédécesseur immédiat dans L (respectivement prédécesseur de son successeur immédiat dans L) (lorsque de tels éléments existent) (fig. 3).

2.2.7. *Construction des associations unitaires*

La dernière étape de la méthode consiste à transcrire le contenu spécifique des cliques dans une matrice d'incidence «cliques - espèces». Cas triviaux mis à part, une telle matrice n'a pas la propriété des 1-consécutifs. Pour la lui donner il suffit de remplacer par des 1 les zéros qui sont intercalés entre des 1 dans les colonnes. Seules les lignes de la matrice résultante qui correspondent à des cliques maximales sont finalement conservées (les autres sont radiées): chaque ligne de cette dernière matrice compactée correspond à une association unitaire et le graphe qui lui est associé est un graphe d'intervalles.

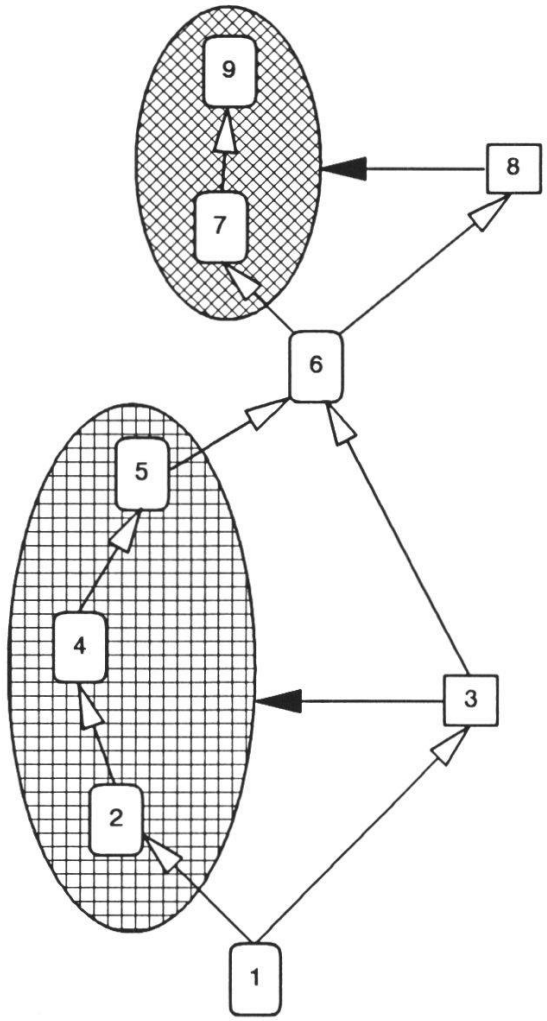

Figure 3.-Réduction à un chemin unique L'. Les ellipses indiquent les ensembles auxquels le test de best-fit est appliqué pour les cliques 3 et 8 qui ne sont pas contenues par le chemin maximal L de Gk.

2.2.8. *Arêtes virtuelles résiduelles*

La procédure décrite ci-dessus a pour effet de ne pas prendre en compte les arêtes virtuelles (i.e. $i \rightarrow j$ et $j \rightarrow i$) dans la phase initiale du calcul. Ceci tient au fait que (par définition) les couples d'espèces qui correspondent à ces arêtes ne sont observés dans aucun horizon maximal. En général ces arêtes sont générées artificiellement lors de la transformation de G* en un graphe d'intervalles. Lorsque ça n'est pas le cas, une routine de détection permet de les insérer dans la matrice M lors de la phase final du calcul.

Une arête résiduelle est détectée lorsqu'il existe un arc $x_j \rightarrow x_i$ alors que x_j se trouve dans une AU (k_{k+n}) située au-dessus de celle contenant x_i (k_k) (fig. 4 a).

Le critère déterminant la méthode de réinsertion repose sur la préservation maximale des arcs existant entre x_i (respectivement x_j) et ses successeurs (respectivement prédécesseurs) présents dans l'intervalle k_k - k_{k+n} qui sont déterminés par les relations suivantes:

$$A = (k_{k+1} \cup ... \cup k_{k+n}) - k_k \text{ et } B = (k_k \cup ... \cup k_{k+n-1}) - k_{k+n}$$

La somme des reproductibilités des arcs $x_i \rightarrow A$ et $B \rightarrow x_j$ donnent deux valeurs V(A) et V(B) qui permettent de choisir entre les trois solutions suivantes pour réinsérer l'arête (x_i, x_j) (fig. 4):

-si V(A)>V(B) alors x_j est ajouté à l'intervalle $k_{k+n-1}..k_{k+1}$ (fig. 4 b);

-si V(A)<V(B) alors x_i est ajouté à l'intervalle $k_{k+1}..k_{k+n}$ (fig. 4 c);

-si V(A)=V(B) alors x_i est ajouté à $k_{k+1}..k_{k+n/2}$ et x_j à $k_{k+n-1}..k_{k+n/2}$ (fig. 4 d).

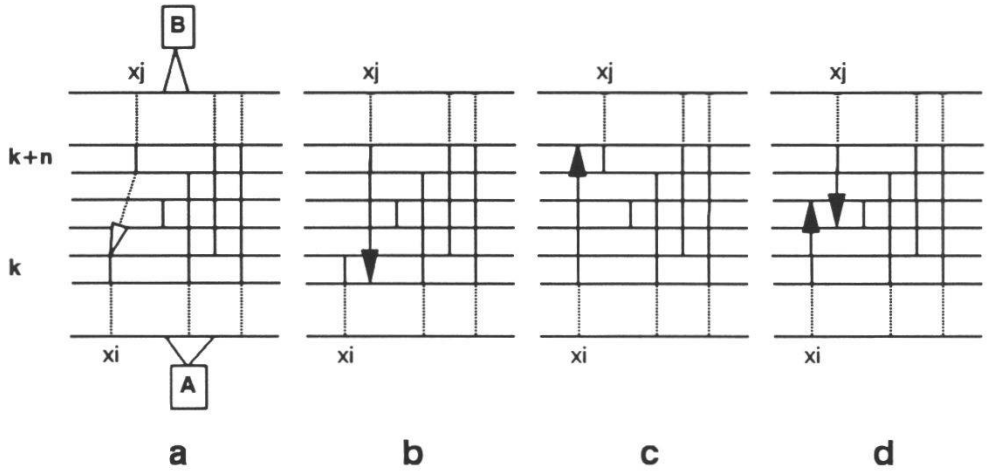

Figure 4.-Insertion des arêtes virtuelles résiduelles.

2.2.9. *Arcs résiduels*

Certaines superpositions stratigraphiques conflictuelles détruites lors de la transformation de G* peuvent être restituées à la fin du processus. Cette étape sert à accroître la résolution du protoréférentiel. La récupération de ces arcs résiduels se fait de la façon suivante.

Dans chaque association unitaire k_i du premier protoréférentiel calculé par le programme (i.e. construit immédiatement après la compaction de la matrice associée à L'), sont définis deux ensembles d'espèces, A et D, où A est l'ensemble des espèces qui apparaissent dans k_i et où D est l'ensemble des espèces qui disparaissent dans k_i (fig. 5 a).

S'il existe un arc $x \rightarrow y$ entre un élément x de D et un élément y de A, on dédouble k_i en k_i et k_j (initialement $k_i = k_j$). On retire ensuite l'espèce x de k_j et l'espèce y de k_i (fig. 5 b).

S'il existe plusieurs arcs distincts de $x \rightarrow y$, la même opération est répétée pour chaque arc en vérifiant à chaque pas que le retrait de x de k_j (respectivement y de k_i) n'aboutit pas à la destruction d'une arête réelle ou virtuelle (p.ex. (x, z) ou (y, z)) (fig. 5 b). Dans un tel cas le retrait n'est pas effectué (fig. 5 c).

Lorsque cette opération a été appliquée à toute les associations unitaires, le test de maximalité est refait et les ensembles non maximaux sont supprimés.

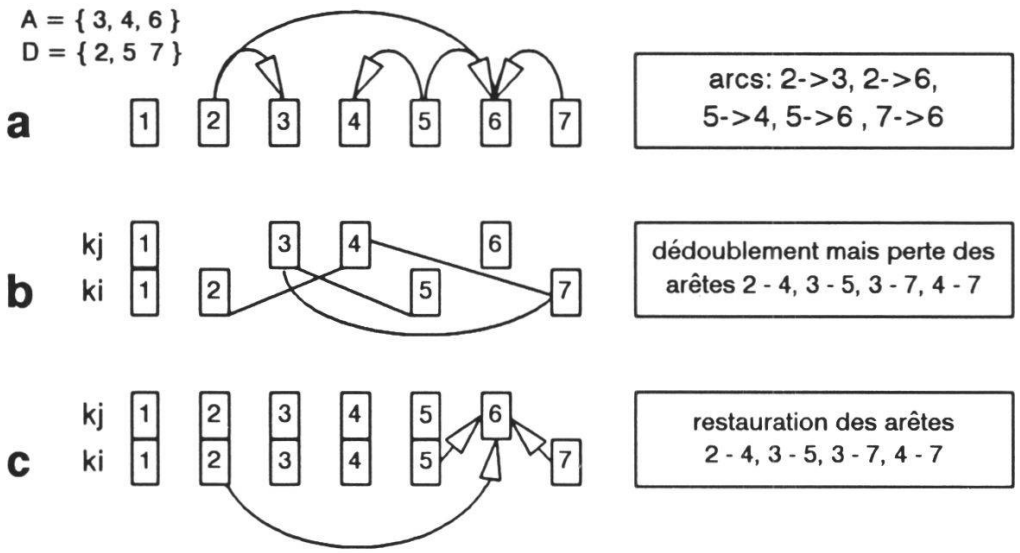

Figure 5.-Insertion des arcs résiduels.

2.2.10. *Identification des associations unitaires et corrélations*

La dernière opération effectuée par notre programme consiste à construire les tableaux de corrélations entre les profils à partir du tableau des associations unitaires. Cette procédure est réalisée de la façon suivante.

Notons X_{ij} le contenu spécifique du $i^{ème}$ niveau du $j^{ème}$ profil.

Notons $K = \{k_m, m=1 \text{ à } p\}$ l'ensemble des associations unitaires obtenues.

Le calcul suivant est répété pour chaque niveau de chaque profil:

$$X_{ij} - k_m = I$$

Si $I = \varnothing$, le niveau considéré est assigné à l'association unitaire (ou aux associations unitaires) correspondante(s).

3. Aspects informatiques de BioGraph

3.1. *Introduction*

Le développement d'un logiciel basé sur les idées algorithmiques exposées plus haut nécessite préalablement une définition claire des modalités du programme: langage, compilateur, environnement de développement, structures de données, librairies, récursivité, etc.

La nature du problème à traiter implique l'usage d'un ensemble modulaire de routines, la taille des données à manipuler suppose une gestion dynamique de la mémoire et le nombre des itérations nécessite une très grande vitesse

d'exécution. C'est le compilateur *TopSpeed Modula-2* de JENSEN et PARTNERS qui répond le mieux à nos exigences et c'est ainsi que nous avons construit le cœur de notre programme en langage *Modula-2*. Une partie des programmes de service de BIOGRAPH a été écrite en *Turbo Pascal* de BORLAND INTERNATIONAL.

La totalité du programme a été développée sur un micro-ordinateur IBM AT et testé avec succès sur toute une gamme de machines compatibles IBM, portables ou non, et dotées de processeurs 8088, 80286 ou 80386.

L'utilisateur bénéficie d'un menu interactif convivial qui lui permet de gérer les différentes options de calcul, d'accéder à l'éditeur de son choix pour rédiger ses données, ou encore de profiter d'un module graphique de représentation des différents graphes.

3.2. *Architecture du programme*

Pour marier le *Modula-2* avec le *Turbo Pascal* et pour minimiser la taille du code, BioGraph est segmenté en plusieurs programmes et fichiers (fig. 6) qui s'articulent autours de BG.EXE, écrit en *Turbo Pascal*. Ce dernier, après contrôle de la présence de tous les fichiers nécessaires, lance et gère les appels et les réponses des différents éléments. La communication se fait par l'intermédiaire d'une zone de mémoire réservée et accessible à tous les programmes.

Les programmes invoqués sont les suivants:

-BG_MENU.EXE (*Turbo Pascal*) sert d'interface avec l'utilisateur. Il s'agit d'un menu interactif avec des explications intégrées dans le fichier d'aide BG.HLP. Il permet de configurer le mode de calcul du graphe, de choisir le fichier de données et d'exécuter diverses options comme le calcul, la représentation, la conversion et l'édition du graphe et des fichiers. Les sélections faites sont enregistrées dans BG.CFG.

-BG_DATA.EXE (*Turbo Pascal*) analyse la syntaxe et la cohérence du fichier de données au format texte et le traduit sous forme binaire.

-BG_GRAP.EXE (*Modula-2*) procède à l'analyse du graphe et engrange tous les résultats, intermédiaires et finaux, dans des fichiers binaires.

-BG_CONV.EXE (*Turbo Pascal*) traduit et présente sous forme texte ou tableau les fichiers désirés engendrés par BG_GRAP. Les formats sont adaptés aux différents types d'imprimantes.

-BG_DRAW.EXE (*Turbo Pascal*) représente les graphes G*, Gk et Gk'. Son fonctionnement nécessite un écran de type EGA/VGA et une souris. Il permet une manipulation didactique des graphes. Ceux-ci peuvent alors être dessinés sur une imprimante graphique ou un plotter.

Enfin, pour écrire les fichiers de données et les corriger, BIOGRAPH peut recourir à l'éditeur préféré de l'utilisateur. Celui-ci doit être indiqué lors de l'installation de BIOGRAPH qui se fait grâce à INSTALL.EXE. Ce dernier programme se charge de créer le répertoire destiné à BioGraph, d'y copier tous les fichiers nécessaires et de les configurer en fonction de l'environnement.

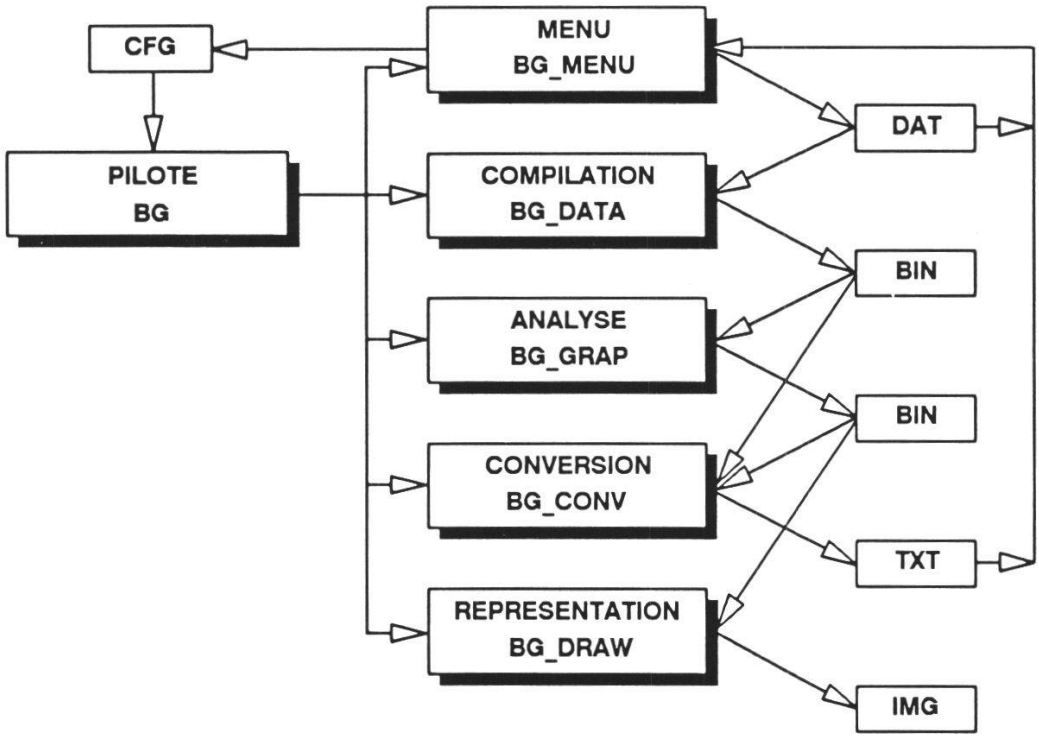

Figure 6.-Organisation de BioGraph.

3.3. *Fichiers*

3.3.1. *Types des fichiers*

Les fichiers sont de deux types: binaire ou texte.

Dans la première catégorie se trouvent tous les fichiers conservant les résultats intermédiaires et finaux.

Dans la seconde catégorie se regroupent les fichiers d'extension:

-DAT, les données lues par BG_DATA;

-TG?, les résultats écrits par BG_CONV;

-REP, le rapport sur les caractéristiques du problème traité.

Les lettres A à M se trouvant à la fin de l'extension TG servent à identifier le type de résultats contenus par les fichiers. Cette notation est expliquée dans le menu interactif de BG_MENU.

Le grand nombre de fichiers et la taille de certains des résultats impliquent l'usage d'un disque dur performant.

3.3.2. *Format des données*

Les données à traiter sont rédigées dans un fichier texte obéissant à une syntaxe souple afin de rendre le codage le plus proche possible des habitudes des utilisateurs. Contrairement à certains programmes, il n'est pas nécessaire de recourir à un encolonnement rigoureux des informations et celles-ci peuvent être aussi bien de type numérique que alphabétique. Quelques mots clefs judicieusement insérés dans les données indiquent au programme comment il doit lire les données (fig. 7). Ce fichier peut être conçu à l'aide de l'éditeur favori de l'utilisateur qui est invoqué depuis BIOGRAPH.

Chaque profil doit être déclaré par un en-tête contenant son nom, codé sur 25 caractères alphanumériques, suivi par les numéros du premier et du dernier niveau de la section. Le contenu fossilifère du profil est décrit à la suite de l'en-tête. Les taxons sont codés sur 5 caractères alphanumériques et leur nombre maximal autorisé est 500.

Deux formats de données sont acceptés:

-le premier, *DATUM*, indique pour chaque taxon les numéros des niveaux d'apparition et de disparition du taxon (fig. 7 a);

-le second, *SAMPLES*, décrit le contenu de chaque horizon (fig. 7 b).

Si nécessaire, une option permet la traduction d'un format à l'autre.

3.4. *Solutions informatiques*

Bien que l'ensemble de ces programmes représente quelques 14'000 lignes de *Pascal* et de *Modula-2*, le cœur du problème repose dans un module de 500 lignes qui contient les définitions de type et toutes les routines de gestion de la mémoire et d'opérations sur les ensembles et les graphes.

3.4.1. *Structures de données*

La première préoccupation d'un informaticien est la définition des structures de données. Lorsque l'analyse de celles-ci est bien conduite, l'écriture du programme se fait naturellement.

Une première difficulté nous est apparu d'emblée, à savoir la taille des données à manipuler. Le soucis de la contraction des informations implique l'usage de structures de données particulières. La représentation de telle ou telle espèce dans un horizon ou une clique ne peut se faire avec des variables de type booléen, numérique ou complexe (liste) sans une perte considérable de place.

Comme le contenu fossilifère d'un horizon se définit par la présence ou l'absence de chaque taxon, il suffit alors de réserver le nombre nécessaire de *words* (mot-machine de 16 *bits*) et d'y valuer chaque *bit*. Il s'agit d'un ensemble ordonné de taille fixe quelque soit le contenu de l'horizon ou de la clique; ordonné dans le sens où la position de chaque *bit* est invariablement attribuée à la même espèce. En outre pour les besoins du problème, chaque ensemble possède un en-tête qui contient un identificateur d'une part, la cardinalité de l'ensemble d'autre part, i.e. le nombre d'éléments appartenant à

A: format DATUM

```
{ Data from DROBNE, K., 1977: Alvéolines paléogènes de l'Istrie et de la
Slovénie. Mem. Suisses Pal., 99, p.1-75. }

DATUM

TITLE
"Short example for BioGraph 2.00: Alvéolines paléogènes de Drobne (1977)"

SECTION Fatji·hrib : BOTTOM 1 - TOP 4
 02ara: 1-1; 03sol: 4-4; 04glo: 3-3; 06pis: 2-2; 07pas: 3-3; 08leu: 4-4;
 12spy: 3-3

SECTION Dane·Divaca: BOTTOM 1 - TOP 4
 01mou: 3-3; 03sol: 1-1; 05ave: 1-1; 06pis: 2-3; 07pas: 1-3; 09mon: 3-4;
 11ded: 3-3; 12spy: 3-3; 13lax: 2-2; 14gui: 4-4

SECTION Veliko: BOTTOM 1 - TOP 7
 02ara: 1-1; 03sol: 2-2; 06pis: 2-3; 07pas: 2-3; 09mon: 6-7; 10arg: 7-7;
 11ded: 5-5; 12spy: 4-4; 13lax: 3-3; 14gui: 6-7; 15dec: 4-6

SECTION Ritomece: BOTTOM 1 - TOP 4
 01mou: 1-2; 04glo: 3-3; 07pas: 1-1; 08leu: 2-2; 09mon: 1-4; 10arg: 3-4;
 11ded: 1-2; 12spy: 2-2; 14gui: 3-4; 15dec: 4-4

SECTION Podgorje: BOTTOM 1 - TOP 2
 04glo: 2-2; 05ave: 1-1

SECTION Podgrad·Hrusica: BOTTOM 1 - TOP 4
 01mou: 2-2; 04glo: 3-3; 07pas: 1-2; 09mon: 3-3; 10arg: 4-4; 13lax: 1-1;
 14gui: 4-4

SECTION Kozina·Socerb: BOTTOM 1 - TOP 4
 02ara: 1-1; 03sol: 3-3; 05ave: 2-2; 07pas: 3-4; 13lax: 4-4; 15dec: 4-4

SECTION Golez: BOTTOM 1 - TOP 6
 02ara: 1-1; 05ave: 1-1; 06pis: 3-3; 07pas: 2-2; 08leu: 5-5; 10arg: 6-6;
 13lax: 4-4; 14gui: 5-6

SECTION Zbernica: BOTTOM 1 - TOP 2
 06pis: 2-2; 10arg: 1-2; 12spy: 2-2; 15dec: 1-1

SECTION Dane·Istrie: BOTTOM 1 - TOP 3
 01mou: 1-1; 06pis: 1-1; 08leu: 3-3; 10arg: 2-2; 11ded: 1-1; 12spy: 1-1

SECTION Jelsane: BOTTOM 1 - TOP 1
 09mon: 1-1; 10arg: 1-1; 14gui: 1-1
```

B: format SAMPLES (début des mêmes données)

```
SAMPLES

SECTION Fatji·hrib, BOTTOM 1, TOP 4
 < 4: 03sol, 08leu
 < 3: 04glo, 07pas, 12spy
 < 2: 06pis
 < 1: 02ara
```

Figure 7.-Exemple de fichier de données.

l'ensemble. La déclaration donnée dans la figure 8 respecte la syntaxe du *Modula-2*.

La déclaration non contrainte du champ de la structure concernant le contenu de l'ensemble permet de jouer sur les propriétés respectives des set et des tableaux sans dimension. L'utilisation d'un tel aggrégat doit obligatoirement passer par un pointeur et les allocations se faire par blocs calculés.

```
EN_TETE    = RECORD
               Index_1, Index_2, Cardinal : CARDINAL
             END;

ENSEMBLE   = RECORD
               En_Tete : EN_TETE;
               CASE : BOOLEAN OF
               | TRUE   : Set   : SET OF CARDINAL
               | FALSE  : Bit   : ARRAY CARDINAL OF BITSET
               END
             END;
POINTEUR   = POINTER TO ENSEMBLE
```

Figure 8.-Définition en *Modula-2* des ensembles de données.

3.4.2. *Opérateurs*

Cette structure de données (*set*) est non seulement compacte, limitée que par la taille de la mémoire disponible, mais encore elle permet un grand nombre d'opérations très rapides sur les ensembles. En effet, tel que nous avons posé l'algorithme, il s'est rapidement avéré que la formalisation ensembliste est de loin la plus performante. Les opérations de base telles union, différence, intersection et différence symétrique se réalisent aisément grâce aux opérateurs binaires présents dans la plupart des langages de programmation. Le *Modula-2* se prête particulièrement bien à ces manipulations car il fournit tous les opérateurs nécessaires, y compris ceux qui permettent l'inclusion et l'exclusion d'un élément dans un ensemble et le test d'appartenance d'un élément à un ensemble. Les informations sont traitées par groupes dont la taille dépend du processeur équipant l'ordinateur. Il y a donc moins d'itérations à effectuer qu'avec une autre structure de données. Il faut évidemment maintenir à jour la cardinalité de chaque ensemble afin de reconnaître rapidemment un ensemble vide.

3.4.3. *Pointeurs*

Pour opérer sur de grands lots d'informations, de taille très variable, il est nécessaire de procéder à une gestion dynamique de la mémoire vive. Pour ce faire il faut posséder de nombreuses routines d'allocation et de libération des blocs de mémoire, et de calcul des pointeurs associés, spécifiques aux types définis plus haut. Notre programme fonctionnant sous DOS, nous avons dû contourner le problème de la segmentation de la mémoire en blocs de 64 kb.

Dans le cas des ensembles, il faut recourir à deux types de pointeurs et de calcul de ceux-ci: le premier sur l'en-tête de l'ensemble, le second se mouvant sur chaque élément à l'intérieur de l'ensemble lui-même. Avec le type défini précédemment, il s'agit plus précisément d'un calcul d'indice.

A plusieurs étapes de l'algorithme, des matrices doivent être gérées, comme la matrice associée au graphe biostratigraphique et celle du graphe Gk. Ce sont elles qui encombrent le plus la mémoire, car chaque cellule est un entier dont le codage permet d'évaluer les relations du graphe. Une routine particulière d'indexation limite l'accès à la demi-matrice, seule partie nécessaire, divisant donc par deux la place utilisée.

3.4.4. *Exemples de procédure*

En guise d'exemple, nous avons sélectionné deux procédures représentatives de notre travail, qui n'ont pas fait l'objet d'une description théorique au chapitre 2. Elles sont présentées en un pseudo-langage se rapprochant quelque peu du *Pascal*. Les mots clefs définissant les structures de contrôle sont en caractères gras, les variables en minuscules, les constantes et les types en capitales, les noms des fonctions ou des procédures explicites débutent par une majuscule. Les opérations sur les ensembles, en notation mathématique, et les variables associées sont effectivement remplacées dans le programme par des procédures et fonctions plus sophistiquées.

3.4.4.1. *Réduction*

Il existe une routine simple, omniprésente et spécifique à notre problème: la réduction. Elle illustre l'emploi des ensembles, tels qu'ils ont été définis plus haut et elle est représentative du style de programmation appliqué aux autres opérations. A chaque étape, il est souhaitable de réduire le nombre d'objets à traiter, autrement dit la réduction ne conserve que les informations significatives. Elle est utilisée dans la recherche des horizons maximaux locaux et résiduels, et par la suite elle permet de compacter les cliques et les associations unitaires. C'est ainsi une routine élémentaire d'élimination des ensembles non maximaux (fig. 9).

Les ensembles sont d'abord triés par cardinalité décroissante en recourant par exemple à la technique classique du *quicksort*. Puis le plus grand ensemble de la liste est comparé aux suivants en appliquant le test de différence qui met en évidence les sous-ensembles. Cette approche est beaucoup plus rapide que celle qui consiste à chercher si chaque élément du

premier ensemble est contenu dans le second. Chaque sous-ensemble est éliminé et le nombre total d'ensembles est mis à jour. Ce qui est noté «ensemble[i]» remplace le calcul d'un pointeur sur l'en-tête de l'ensemble «i».

```
Tri_des_ensembles_par_cardinalité_décroissante;
premier ⇐ 1;
repeat
  for i ⇐ premier to (nombre_ensembles - 1) do
      if ((ensemble[i + 1] - ensemble[i]) = ∅) then
          ensemble[i + 1] ⇐ ∅;
      }
  }
  premier ⇐ (premier + 1);
  nombre_ensembles ⇐ Elimination_des_ensembles_nuls;
until (premier ≥ nombre_ensembles);
return nombre_ensembles;
```

Figure 9.-Elimination des ensembles non maximaux.

3.4.4.2. Chemin maximal

Dans la littérature consacrée aux algorithmes informatiques une large place est faite aux différentes méthodes de détection du plus court chemin entre deux sommets, justifiée par leurs nombreux domaines d'application. Par contre la recherche du plus long chemin d'un graphe est généralement omise. Nous proposons ici un algorithme applicable aux graphes strictement orientés (sans composante fortement connexe).

L'idée centrale de cet algorithme repose sur un détournement de la recherche en profondeur (*depth-first-search*) abondamment décrite dans la littérature informatique. Contrairement à la recherche en profondeur classique, nous autorisons sous certaines conditions le parcours d'une branche déjà examinée de l'arbre de recherche. Comme cet algorithme est récursif, nous utilisons à la fois les propriétés de «descente» et de «remontée».

Lors du parcours en profondeur à partir d'un sommet, le niveau (profondeur ou hauteur) à partir du sommet initial (racine) est attribué au sommet inspecté (fig.10). Il y a deux conditions d'interruption des appels récursifs:

-l'absence de successeur indique une extrémité (feuille) de l'arbre de recherche;

-le rang du successeur est plus élevé, autrement-dit il a déjà été considéré dans un parcours plus long.

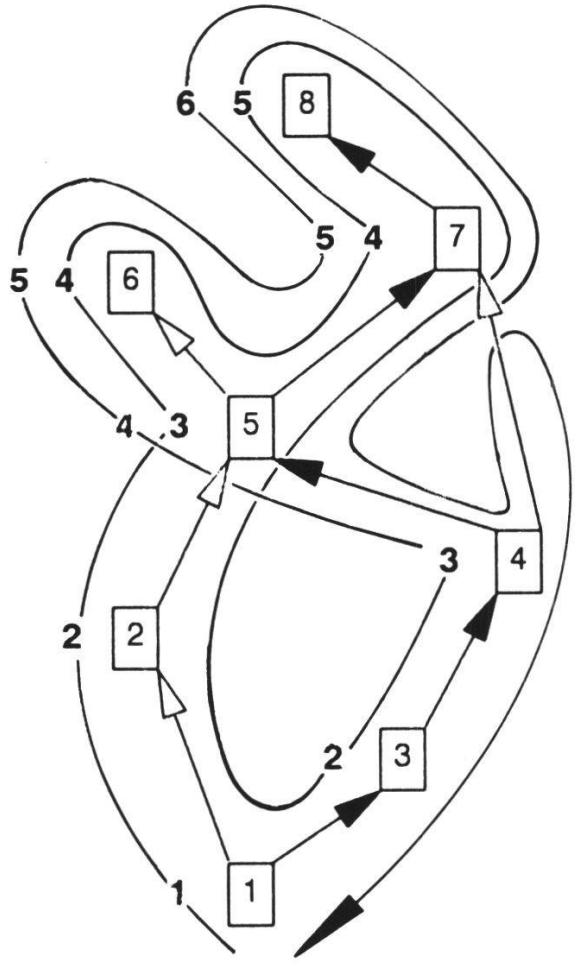

Figure 10.-Hauteur des sommets lors d'un parcours en profondeur.

Lorsqu'une extrémité est atteinte, on regarde si elle correspond à un niveau plus élevé que celles qui ont déjà été examinées. Si c'est le cas, il s'agit de la terminaison du plus long chemin de la partie du graphe considérée jusque là et les sommets parcourus sont alors enregistrés dans une liste lors du retour vers la racine.

Au départ le niveau de chaque sommet et la longueur du chemin maximal sont nulles. Cette routine (fig. 11) doit être appliquée à tous les sommets sans prédécesseur, c'est-à-dire à tous les sommets racines susceptibles d'être à l'origine du plus long chemin. Cet algorithme peut aisément être modifié pour chercher le plus long chemin entre deux sommets en ajoutant une condition de sortie lorsque le sommet terminal est atteint. De même, lorsqu'il existe plusieurs chemins maximaux, il est possible d'ajouter un critère de choix basé sur le poids des arcs ou des sommets parcourus. De nombreuses optimisations peuvent également être ajoutées afin de limiter les parcours et les itérations.

```
Maximal( noeud, niveau          : integer;
         var hauteur, chemin    : array;
         var longueur           : integer ) : boolean;

var  successeur                 : integer;
     chemin_maximal, feuille    : boolean;

niveau ⇐ niveau + 1;
hauteur[noeud] ⇐ niveau;
feuille ⇐ VRAI;
chemin_maximal ⇐ FAUX;
for successeur ⇐ 1 to nombre_noeuds do
  if (hauteur[successeur] ≤ niveau)
  and (successeur ∈ Γ°+(noeud)) then
      feuille ⇐ FAUX;
      if Maximal(successeur, niveau, hauteur, chemin, longueur)
      = VRAI then
          chemin_maximal ⇐ VRAI;
      }
  }
}
if (feuille = VRAI) and (longueur < niveau) then
  longueur ⇐ niveau;
  chemin_maximal ⇐ VRAI;
}
if chemin_maximal = VRAI then
  chemin[niveau] ⇐ noeud;
}
return chemin_maximal;
```

Figure 11.-Recherche du chemin maximal. (Le graphe et son nombre de sommets (*nombre_noeuds*) sont des variables globales.)

4. EXEMPLE D'APPLICATION DE BIOGRAPH

La meilleure façon de décrire les différentes étapes du programme consiste à partir d'un exemple d'application simple. A cette fin, nous reprendrons ici l'analyse d'un problème standard (les alvéolines ilerdiennes de DROBNE 1977) dont les données de base sont compilées dans la figure 7. Toutes les figures qui illustrent notre exemple sont produites par BIOGRAPH.

4.1. *Horizons maximaux et graphe G**

La première étape du calcul consiste à construire les horizons maximaux locaux et résiduels: ce premier résultat est donné dans les figures 12 et 13. Les horizons résiduels sont notés k et les indices i et j signifient «j^{ème} horizon du i^{ème} profil».

Le graphe biostratigraphique G* est établi à partir de ces données. Ce graphe est représenté sous la forme de deux listes distinctes: la première recense l'ensemble de ses arêtes (fig. 14 a) et la deuxième énumère l'ensemble de ses arcs avec leur reproductibilité respective (fig. 14 b); exemple: l'arc (01MOU → 04GLO) a une reproductibilité égale à 2.

```
Section FATJI·HRIB

4 [2]: 03SOL 08LEU
3 [3]: 04GLO 07PAS 12SPY
2 [1]: 06PIS
1 [1]: 02ARA

Section DANE·DIVACA

4 [2]: 09MON 14GUI
3 [6]: 01MOU 06PIS 07PAS 09MON
       11DED 12SPY
2 [3]: 06PIS 07PAS 13LAX
1 [3]: 03SOL 05AVE 07PAS

Section VELIKO

7 [3]: 09MON 10ARG 14GUI
6 [3]: 09MON 14GUI 15DEC
5 [2]: 11DED 15DEC
4 [2]: 12SPY 15DEC
3 [3]: 06PIS 07PAS 13LAX
2 [3]: 03SOL 06PIS 07PAS
1 [1]: 02ARA

Section RITOMECE

4 [4]: 09MON 10ARG 14GUI 15DEC
3 [4]: 04GLO 09MON 10ARG 14GUI
2 [5]: 01MOU 08LEU 09MON 11DED
       12SPY
1 [4]: 01MOU 07PAS 09MON 11DED

Section PODGORJE

2 [1]: 04GLO
1 [1]: 05AVE

Section PODGRAD·HRUSICA

4 [2]: 10ARG 14GUI
3 [2]: 04GLO 09MON
2 [2]: 01MOU 07PAS
1 [2]: 07PAS 13LAX

Section KOZINA·SOCERB

4 [3]: 07PAS 13LAX 15DEC
3 [2]: 03SOL 07PAS
2 [1]: 05AVE
1 [1]: 02ARA

Section GOLEZ

6 [2]: 10ARG 14GUI
5 [2]: 08LEU 14GUI
4 [1]: 13LAX
3 [1]: 06PIS
2 [1]: 07PAS
1 [2]: 02ARA 05AVE

Section ZBERNICA

2 [3]: 06PIS 10ARG 12SPY
1 [2]: 10ARG 15DEC

Section DANE·ISTRIE

3 [1]: 08LEU
2 [1]: 10ARG
1 [4]: 01MOU 06PIS 11DED
12SPY

Section JELSANE

1 [3]: 09MON 10ARG 14GUI
```

Figure 12.-Horizons maximaux locaux.

k2.3	[6]:	01MOU	06PIS	07PAS	09MON	11DED	12SPY
k4.2	[5]:	01MOU	08LEU	09MON	11DED	12SPY	
k4.3	[4]:	04GLO	09MON	10ARG	14GUI		
k4.4	[4]:	09MON	10ARG	14GUI	15DEC		
k2.2	[3]:	06PIS	07PAS	13LAX			
k9.2	[3]:	06PIS	10ARG	12SPY			
k7.4	[3]:	07PAS	13LAX	15DEC			
k1.3	[3]:	04GLO	07PAS	12SPY			
k2.1	[3]:	03SOL	05AVE	07PAS			
k3.2	[3]:	03SOL	06PIS	07PAS			
k1.4	[2]:	03SOL	08LEU				
k3.5	[2]:	11DED	15DEC				
k8.1	[2]:	02ARA	05AVE				
k3.4	[2]:	12SPY	15DEC				
k8.5	[2]:	08LEU	14GUI				

Figure 13.-Horizons maximaux résiduels.

A. Arêtes de G^*

01MOU	: 06PIS	07PAS	08LEU	09MON	11DED	12SPY
02ARA	: 05AVE					
03SOL	: 05AVE	06PIS	07PAS	08LEU	12SPY	
04GLO	: 07PAS	08LEU	09MON	10ARG	12SPY	14GUI
05AVE	: 07PAS					
06PIS	: 07PAS	09MON	10ARG	11DED	12SPY	13LAX 15DEC
07PAS	: 09MON	11DED	12SPY	13LAX	15DEC	
08LEU	: 09MON	10ARG	11DED	12SPY	14GUI	
09MON	: 10ARG	11DED	12SPY	14GUI	15DEC	
10ARG	: 12SPY	14GUI	15DEC			
11DED	: 12SPY	15DEC				
12SPY	: 15DEC					
13LAX	: 15DEC					
14GUI	: 15DEC					

B. Arcs (→) de G^*

01MOU	: 04GLO	[2]	10ARG	[3]	14GUI	[3]	15DEC	[1]			
02ARA	: 03SOL	[3]	04GLO	[1]	06PIS	[3]	07PAS	[4]	08LEU	[2]	
	09MON	[1]	10ARG	[2]	11DED	[1]	12SPY	[2]	13LAX	[3]	
	14GUI	[2]	15DEC	[2]							
03SOL	: 01MOU	[1]	09MON	[2]	10ARG	[1]	11DED	[2]	13LAX	[3]	
	14GUI	[2]	15DEC	[2]							
04GLO	: 03SOL	[1]	15DEC	[1]							
05AVE	: 01MOU	[1]	04GLO	[1]	06PIS	[2]	08LEU	[1]	09MON	[1]	
	10ARG	[1]	11DED	[1]	12SPY	[1]	13LAX	[3]	14GUI	[2]	
	15DEC	[1]									
06PIS	: 04GLO	[1]	08LEU	[3]	14GUI	[3]					
07PAS	: 08LEU	[3]	10ARG	[4]	14GUI	[5]					
08LEU	: 15DEC	[1]									
11DED	: 04GLO	[1]	10ARG	[3]	14GUI	[3]					
12SPY	: 14GUI	[3]									
13LAX	: 01MOU	[2]	04GLO	[1]	08LEU	[1]	09MON	[3]	10ARG	[3]	
	11DED	[2]	12SPY	[2]	14GUI	[4]					

Figure 14.-Relations dans le graphe biostratigraphique.

4.2. *Constructions des cliques maximales*

La deuxième étape du calcul consiste à réunir les horizons résiduels dont les éléments forment une clique (fig. 15).

Dans l'exemple étudié ici, tous les éléments de k7.4 sont compatibles avec ceux de k2.2. k7.4 et k2.2 sont donc réunis. k8.5 et k4.3 présentent la même situation. Les horizons dont les contenus sont réunis sont notés k*.

```
k2.3   [6]:01MOU   06PIS   07PAS   09MON   11DED   12SPY
k3.5*  [6]:06PIS   07PAS   09MON   11DED   12SPY   15DEC
k8.5*  [5]:04GLO   08LEU   09MON   10ARG   14GUI
k4.2   [5]:01MOU   08LEU   09MON   11DED   12SPY
k9.2*  [5]:06PIS   09MON   10ARG   12SPY   15DEC
k3.2*  [4]:03SOL   06PIS   07PAS   12SPY
k2.2*  [4]:06PIS   07PAS   13LAX   15DEC
k4.4   [4]:09MON   10ARG   14GUI   15DEC
k1.3*  [4]:04GLO   07PAS   09MON   12SPY
k2.1   [3]:03SOL   05AVE   07PAS
k1.4*  [3]:03SOL   08LEU   12SPY
k8.1   [2]:02ARA   05AVE
```

Figure 15.-Cliques maximales.

4.3. *Construction de la matrice M et du graphe Gk*

L'étape suivante consiste à établir les relations entre les cliques. Les relations de la matrice M sont données dans la figure 16. La signification des symboles utilisés dans cette figure a été exposée plus haut.

Exemples:

-la relation entre k2.3 et k3.5* n'est pas conflictuelle (V(A)=2 et V(B)=0);

-la relation entre k2.3 et k2.2* est conflictuelle (V(A)=2 et V(B)=13);

-la relation entre k2.3 et k1.4* est indéterminée (V(A)=V(B)=8).

```
2.3   : ⇒ 3.5*  ⇒ 8.5*  ⇒ 4.2   ⇒ 9.2*  ⇐ 3.2*  ← 2.2*  ⇒ 4.4
        ⇒ 1.3*  ⇐ 2.1   ⇄ 1.4*  ⇐ 8.1
3.5*  : → 8.5*  → 4.2   ⇒ 9.2*  ⇐ 3.2*  ⇐ 2.2*  ⇒ 4.4   → 1.3*
        ⇐ 2.1   ← 1.4*  ⇐ 8.1
8.5*  : ⇐ 4.2   ← 9.2*  ← 3.2*  ← 2.2*  ⇒ 4.4   ⇐ 1.3*  ← 2.1
        ← 1.4*  ⇐ 8.1
4.2   : → 9.2*  ⇐ 3.2*  ← 2.2*  ⇒ 4.4   → 1.3*  ⇐ 2.1   ⇐ 1.4*
        ⇐ 8.1
9.2*  : ⇐ 3.2*  ⇐ 2.2*  ⇒ 4.4   ← 1.3*  ⇐ 2.1   ← 1.4*  ⇐ 8.1
3.2*  : → 2.2*  ⇒ 4.4   → 1.3*  ⇐ 2.1   ⇒ 1.4*  ⇐ 8.1
2.2*  : ⇒ 4.4   → 1.3*  ⇐ 2.1   → 1.4*  ⇐ 8.1
4.4   : ⇐ 1.3*  ⇐ 2.1   ⇐ 1.4*  ⇐ 8.1
1.3*  : ← 2.1   → 1.4*  ⇐ 8.1
2.1   : ⇒ 1.4*  ⇐ 8.1
1.4*  : ⇐ 8.1
```

Figure 16.-Relations entre les cliques maximales (matrice M associée à Gk).

4.4. *Recherche et destructions des composantes fortement connexes de Gk*

Dans notre exemple, le graphe orienté Gk contient une composante fortement connexe à 4 sommets. Dans cette composante il existe un arc fort (non conflictuel) (k1.4* ⇒ k4.2) et 5 arcs conflictuels. Comme cette composante est constituée par deux cycles C3 imbriqués, il est nécessaire de détruire deux arcs. Ce sera dans l'ordre k4.2 → k1.3* et k1.4* → k3.5* qui présentent les valeurs conflictuelles les plus élevées (fig. 17).

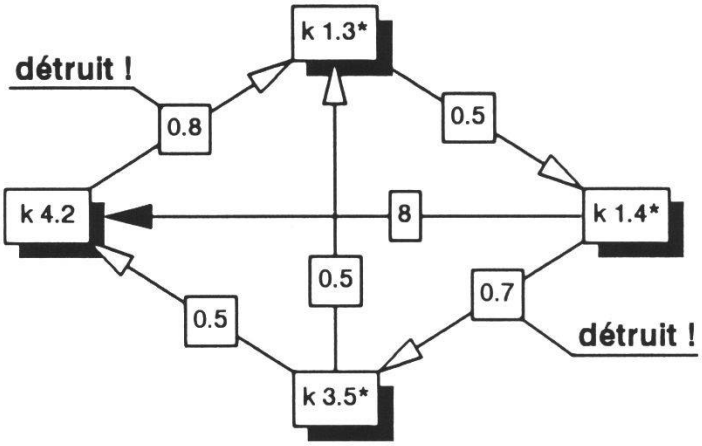

Figure 17.-Composante fortement connexe de Gk.

4.5. *Associations unitaires et corrélations*

Une fois le chemin maximal construit et la réunion par *best-fit* réalisée, nous obtenons les associations unitaires (figure 18 a) résultant de l'élimination des cliques non maximales de la matrice M. Pour satisfaire les besoins du biostratigraphe le programme se charge de faire une représentation des associations unitaires triées par apparition ou disparition. Pour terminer, le programme crée un tableau de corrélation (fig.18 b) indiquant quelles associations unitaires ont été strictement identifiées dans chaque section.

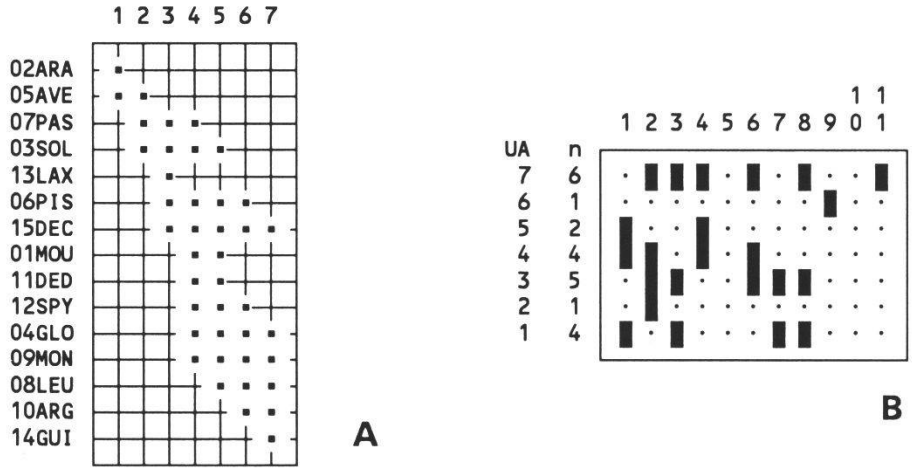

Figure 18.a.-Associations unitaires. b.-Tableau de corrélation.

5. Bibliographie

Carre B., 1979. Graphs and Networks. Oxford Applied Mathematics and Computing Science Series. 277 p.

Drobne K., 1977. Alvéolines paléogènes de l'Istrie et de la Slovénie. *Mém. Suisses Pal. 99*: 1-75.

Guex J., 1987. Corrélations biochronologiques et associations unitaires. Presses polytechniques romandes, Lausanne. 244 p.

Guex J., 1988. Utilisation des horizons maximaux résiduels en biochronologie. *Bull. Labo Géol. Univ. Lausanne 300* et *Bull. Soc. Vaud. Sc. Nat. 79.2*: 135-142.

Guex J. et Davaud E., 1982. Recherche automatique des associations unitaires en biochronologie. *Bull. Lab. Géol. Univ. Lausanne 261* et *Bull. Soc. Vaud. Sc. Nat. 76*: 53-69.

Guex J. et Davaud E., 1984. Unitary Associations Method: Use of Graph Theory and Computer Algorithm. *Comp. Geosc. 10/1*: 69-96.

Roberts F., 1976. Discrete mathematical models. Prentice Hall, New Jersey. 559 p.

Manuscrit reçu le 15 avril 1991