Materials and methods

CompositeSearch algorithm

CompositeSearch is a multithreaded tool, which detects composite genes and their families, as well as the various components, in a set of biological sequences. User must provide an all-against-all BLAST output of these sequences as an input for CompositeSearch.

STEP 1: Construction of the SSN

The SSN is constructed by CompositeSearch, based on the cleaned result of an all-against-all BLAST sequence comparison. This preliminary step relies on a C++ program called *cleanBlastp*, provided along with CompositeSearch. *cleanBlastp* uniquely numbers each sequence in the BLAST output, and removes all self-hits, keeping the best hit (i.e. lowest E-value) amongst multiple hits between pairs of sequences. At the end of this preliminary step, the input file used by CompositeSearch contains BLAST information about matches between pairs of sequences (qstart, qend, sstart, send), sequence length (qlen, slen) and their symmetrized similarity scores (E-value, pident). The selection of unique pairs of hits avoids simultaneous memory access issues and allows to parallelize the SSN construction, by splitting the cleaned BLASTP results file into a user defined number of CPUs. CompositeSearch utilizes user-defined similarity scores (default E-value ≤ 10, default Pident $\geq 30\%$) to construct the SSN. The results are then represented as an undirected network $G=(V,E)$, where V is the set of sequences, and edge is $(u,v) \in E$ if the similarity score S_{uv} or S_{vu} is higher than a user-defined threshold.

STEP 2: Definition of gene families

CompositeSearch clusters sequences into gene families in two steps. First, it uses a modified Depth First Search (DFS) algorithm on a thresholded SSN (default: mutual coverage between two sequences $\geq 80\%$) that defines connected components (CCs). Each CC is considered as a putative gene family, when the minimum mutual sequence coverage criterion is high (≥ 80 %), but gene family definition is then further refined in a second step as follows. Each time a CC is detected a mutual coverage score (S_{mc}) is calculated. If $S_{mc} < 1$, this CC is subjected to the Louvain community detection algorithm (Blondel et al. 2008), using C++ *igraph* 0.7.1 library (Csardi and Nepusz 2006). Indeed, BLAST matches can be over-extended (Mills and Pearson 2013), with the consequence that non-homologous sequences may be introduced in a

CC in pathological cases (Supplementary Figure S1). This second step of community detection allows to define, at a finer granularity, the groups of sequences forming communities (e.g. cliques and/or quasi-cliques) within the CC, which are finally considered as a gene family. Thus each sequence from the original dataset is assigned to a given gene family and a connectivity score is computed for each family. If gene families are pre-computed, a tab delimited file with the gene ID and its family ID can be given as an input and for each of these gene families a connectivity score will be also computed.

This step returns 3 files:

- family.nodes: a file where the nodes for each family is listed which will be useful for post-analysis of the gene families;
- family.edges: a file where the edges for each family is listed;
- family.info: a file storing the number of nodes and edges for each family and their connectivity, which can be used for a sized-based or connectivity-based selection of composite gene families by the user. Connectivity is measured as :

$$C_{family} = \frac{(2 * N_{edges})}{N_{nodes} * (N_{nodes} - 1)}$$

STEP 3: Detection of composite genes

Unlike MosaicFinder and FusedTriplets, CompositeSearch starts from the assumption that each node could be a composite gene. This decision allows to parallelize detection of composite genes by distributing a list of nodes to visit for each CPU, which takes into account node degree to produce computationally balanced lists of nodes to be distributed among the CPUs. CompositeSearch checks whether a node's neighbors belong to different gene families and their size is higher or equal to the minimum number of genes to be used as component gene families. If all neighbors of a node belong to only one gene family, this node is not a composite gene. If at least two neighbors of this node belong to distinct gene families, CompositeSearch takes the sequence corresponding to the node as a reference and maps the matches from all different families along that sequence. Each region with matches from different families along a composite sequence is called a "protein component" hereafter. For each "protein component", CompositeSearch computes an average position for both the start and end of the component (Supplementary Figure S2). If there is no overlap between at least two “protein components” along the reference sequence, then the reference sequence is considered as composite, since it is composed of at least two non-overlapping regions with homology to different gene families. In practice, a maximum overlap can be allowed (by default $\leq$ 20 AA, in order not to discard *bona fide* composite genes despite possible BLAST short overextensions introducing artefactual overlaps between protein components).

During this step, CompositeSearch produces 2 files:

- file.composites : a file in fasta format with the number of the composite sequences and the position and identity of the component families matching along this composite;
- file.compositesinfo: a file containing the number of protein components along a composite sequence, and a non-overlapping score (S_{no}) between all of these components. The S_{no} score is measured as: N_i/N_T, where N_i is the number of non-overlapping pairs of “protein components”, and N_T is the number of all possible pairs of components (N_T). This measure allows the user to sort composite gene families based on the neat separation of all their protein components (S_{no} close to 1) or the separation of some of their protein components only (lower S_{no}) (Supplementary Figure S3).

STEP 4: Detection of composite gene families

CompositeSearch goes through all gene families to check whether a family is composite or not. Any gene family containing at least one composite gene and with a size higher or equal to the minimum number of genes fixed for composite gene family detection is considered as composite family. This process can be parallelized by distributing a list of gene families to analyze for each CPU.

During this step, CompositeSearch produces 2 files:

- file.compositefamilies: a file in fasta format with the number of the composite gene family and the position and identity of the component families matching along this composite;
- file.compositefamiliesinfo: a file containing the connectivity, percentage of composite genes, mean number of “protein components” of the composite gene families.

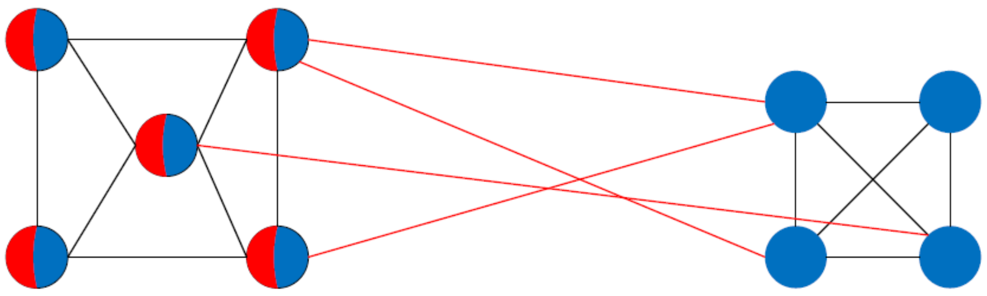
Simulated Data

In order to benchmark CompositeSearch, we simulated families of components and composites (Supplementary Figure S4). We produced gene families with different degrees of divergence as follows. We scaled ultrametric phylogenetic trees with Seq-Gen (option -d) (Rambaut and Grassly 1997) so that the total length of a tree can be measured as the distance from the root to any of the leaves in units of mean number of substitutions per site. We explored 3 evolutionary rates (0.1, 0.5 and 1.0) to cover the range from highly conserved to highly divergent gene families (parameter 1).

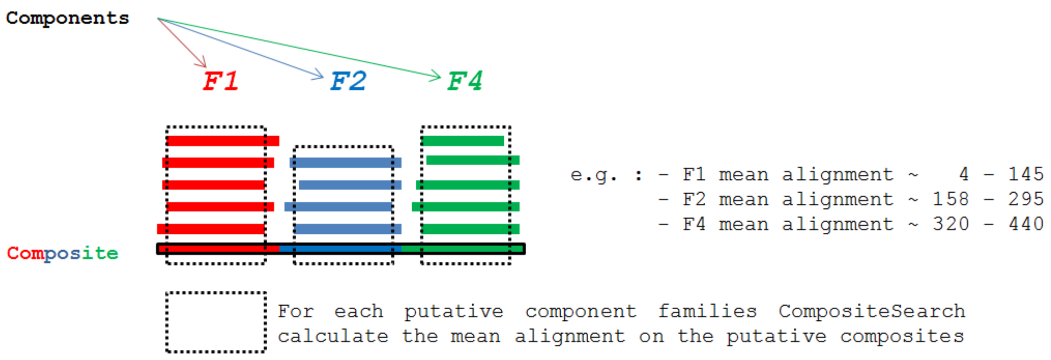
Three component families (A, B and C) have been evolved under the Whelan and Goldman model of amino acid substitution and a site-specific rate heterogeneity following a gamma distribution (alpha=1). Ancestral sequences of 300 amino acids were generated randomly for each component family. These sequences were then evolved at the same evolutionary rate along an ultrametric symmetric binary tree with five depth levels, resulting in component families with 32 genes. These component families will be used for fusion events leading to composite genes with two and three components.

First a pair of sequences *sA* and *sB* is selected from component family A and B at the same depth level *k* from the tree root, from 0 (root) to 5 (tips) (parameter 2) to create a composite sequence with 2 components. Second, a sequence *sC* is selected from component family C at a depth level $p >= k$ from the tree root (from *k* to 5 (parameter 3)) to create composite sequences with 3 components and reassortments.

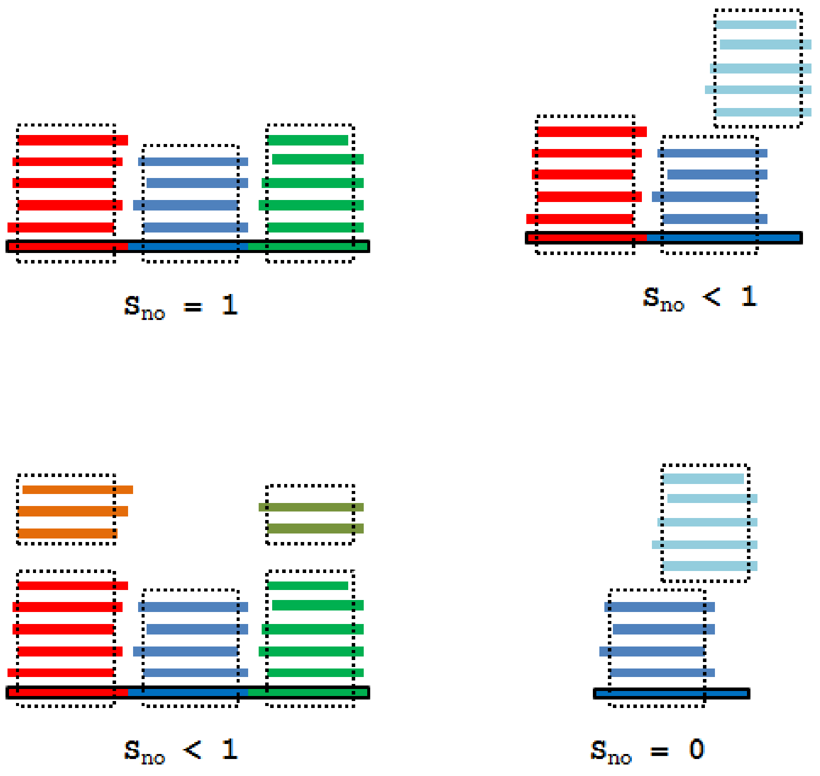
We used *sA* and *sB* to create a novel 300 amino acids composite sequence *sAB* made of 30-50% of the first sequence fused with 70-50% of the second sequence (parameter 4). This ancestral composite sequence *sAB* was then evolved along a perfect binary tree with $q=p-k$ depth levels. This composite family was evolved with evolutionary rates of 0.1, 0.5 or 1 (parameter 5), thereby producing highly conserved to highly divergent composite families. Finally we used an evolved sequence of *sAB* to create three new composite sequences with component reassortments (*sABC*, *sCAB* and *sACB*) made of 30-50% of *sC* (parameter 6). These three composite families were then evolved along a perfect binary tree with $z=5-p$ speciation dates with evolutionary rates of 0.1, 0.5 or 1 (parameter 7). For recent fusion events (k or p = 5), composite sequences were left unmodified. This protocol was repeated 100 times for each combination of the 7 parameters.



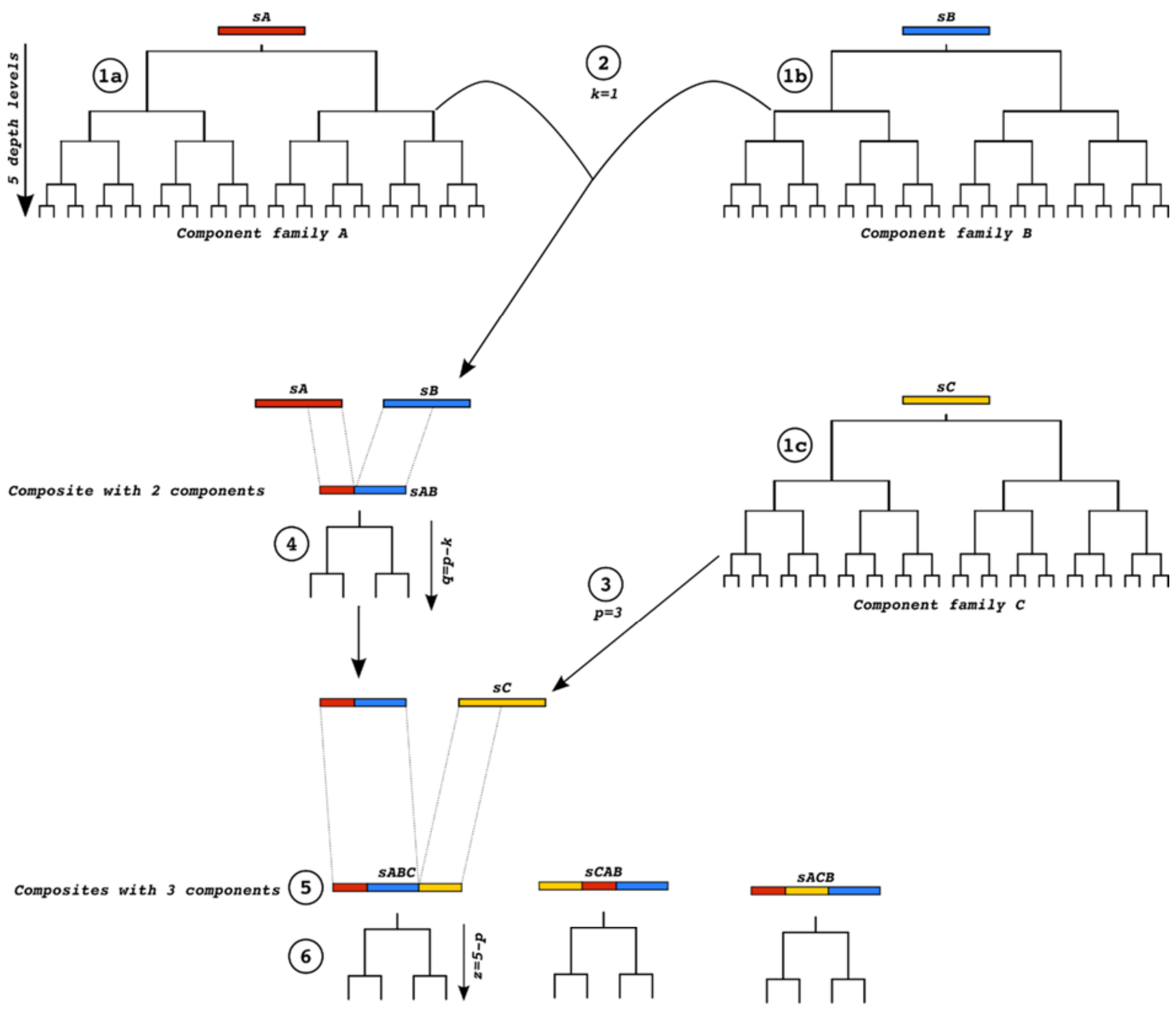
Supplementary Figure S1. Example of a composite gene family (red/blue nodes) and one of its component gene family (blue nodes) detect as one gene family due to BLAST overextensions (red edges). Applying the Louvain clustering will correct the gene families detection by splitting them.


Components
F1
F2
F4
Composite
e.g. : - F1 mean alignment ~   4 - 145
- F2 mean alignment ~ 158 - 295
- F4 mean alignment ~ 320 - 440
For each putative component families CompositeSearch calculate the mean alignment on the putative composites

Supplementary Figure S2. The average position of components' start and end computed by CompositeSearch.



Supplementary Figure S3. Example of non-overlapping score (S_{no}) between all of these components.



1 - Generate 3 random sequences with a length of 300 amino acids
  - Evolve these sequences along a five levels perfect binary tree (1a, 1b and 1c)

2 - Select a pair of sequences (***sA***, ***sB***) along trees 1a and 1b at the same level ***k*** [from 0 to 5]
  - Fuse fragments of ***sA*** [30%-50%] and ***sB*** [70%-50%] sequences given a percentage [30%-50%] to create a composite sequence (***sAB***) of 300 amino acids

3 - Select a sequence ***sC*** along the tree 1c at distance ***p*** >= ***k***

4 - Evolve ***sAB*** along a perfect binary tree of ***q*** level (***q*** = ***p*** - ***k***)

5 - Select an evolved sequence ***sAB***
  - Fuse a fragment of sequence ***sC*** [30%-50%] at different position with ***sAB*** to create composite sequences (***sABC, sCBA and sACB***)

6 - Evolve ***sABC, sCAB*** and ***sACB*** along a perfect binary tree of ***z*** level (***z*** = 5 - ***p***)

Supplementary Figure S4. Simulation and evolution of composite genes.

Evolution Rate			METHODS			
			Composite detection		Composite family detection	
Component	Composite 1	Composite 2	CompositeSearch	FusedTriplets	CompositeSearch	MosaicFinder
0.1	0.1	0.1	99.50	99.50	98.92	95.47
0.1	0.1	0.5	99.48	99.48	98.93	95.43
0.1	0.1	1.0	99.48	99.48	99.03	41.68
0.1	0.5	0.1	99.52	99.52	99.58	95.60
0.1	0.5	0.5	99.54	99.54	99.56	95.59
0.1	0.5	1.0	99.58	99.58	99.58	39.27
0.1	1.0	0.1	99.56	99.56	99.68	95.02
0.1	1.0	0.5	99.62	99.62	99.65	85.06
0.1	1.0	1.0	99.62	99.62	99.67	35.82
0.5	0.1	0.1	99.94	99.94	99.57	98.24
0.5	0.1	0.5	99.94	99.94	99.54	98.34
0.5	0.1	1.0	99.95	99.95	99.61	44.80
0.5	0.5	0.1	99.95	99.95	99.81	98.47
0.5	0.5	0.5	99.95	99.95	99.81	98.33
0.5	0.5	1.0	99.96	99.96	99.81	40.68
0.5	1.0	0.1	99.95	99.95	99.85	97.54
0.5	1.0	0.5	99.95	99.95	99.85	88.03
0.5	1.0	1.0	99.95	99.95	99.87	38.01
1.0	0.1	0.1	99.94	99.94	99.82	99.36
1.0	0.1	0.5	99.96	99.96	99.84	99.37
1.0	0.1	1.0	99.96	99.96	99.86	42.68
1.0	0.5	0.1	99.96	99.96	99.92	99.49
1.0	0.5	0.5	99.96	99.96	99.92	95.94
1.0	0.5	1.0	99.96	99.96	99.93	38.60
1.0	1.0	0.1	99.87	99.87	99.93	91.30
1.0	1.0	0.5	99.89	99.89	99.94	83.99
1.0	1.0	1.0	99.87	99.87	99.93	34.46

Supplementary Table S1. Detection of composite genes and composite gene families on simulated data. This table shows the true positive rate (TPR) for the detection of composite genes (CompositeSearch and FusedTriplets) and the detection of composite gene families (CompositeSearch and MosaicFinder). Depending on the algorithm, CompositeSearch can detect composite genes and composite gene families. For composite detection, TPR is defined as the percentage of genes identified as composite that are indeed composite in the simulation. For composite family detection, TPR is defined as the percentage of genes in the detected composite families that are indeed composite in the simulation. These percentages are computed on 189 possible combinations of parameters explained in the Methods section (component lengths and composite tree levels variation) replicated 100 times averaged over two-component and three-component composites for each of these 27 combinations of evolutionary rates (see Supplementary Figure 4 and Methods section).

Evolution Rate			TPR				FPR
			Composite 1		Composite 2		
Component	Composite 1	Composite 2	EXACT	NON EXACT	EXACT	NON EXACT	
0.1	0.1	0.1	97.83	0.07	90.33	9.60	0.92
0.1	0.1	0.5	97.77	0.14	94.63	5.31	0.87
0.1	0.1	1.0	98.03	0.07	96.32	3.63	0.85
0.1	0.5	0.1	99.11	0.10	88.44	11.51	0.85
0.1	0.5	0.5	99.05	0.12	92.31	7.64	0.86
0.1	0.5	1.0	99.12	0.08	93.73	6.21	0.87
0.1	1.0	0.1	99.35	0.07	86.66	13.28	0.84
0.1	1.0	0.5	99.29	0.07	89.55	10.40	0.83
0.1	1.0	1.0	99.34	0.06	88.38	11.54	0.79
0.5	0.1	0.1	99.12	0.08	92.67	7.26	1.02
0.5	0.1	0.5	98.98	0.16	93.68	6.26	1.12
0.5	0.1	1.0	99.09	0.18	94.17	5.77	1.09
0.5	0.5	0.1	99.54	0.14	90.20	9.74	1.03
0.5	0.5	0.5	99.55	0.12	91.18	8.76	1.10
0.5	0.5	1.0	99.57	0.09	89.96	9.99	1.05
0.5	1.0	0.1	99.65	0.09	88.24	11.70	0.85
0.5	1.0	0.5	99.66	0.09	87.97	11.96	0.91
0.5	1.0	1.0	99.71	0.08	83.28	16.66	0.90
1.0	0.1	0.1	99.63	0.07	88.92	11.00	0.96
1.0	0.1	0.5	99.62	0.11	88.00	11.95	1.05
1.0	0.1	1.0	99.60	0.16	87.00	12.94	0.99
1.0	0.5	0.1	99.77	0.11	86.73	13.21	0.85
1.0	0.5	0.5	99.76	0.13	85.13	14.82	0.88
1.0	0.5	1.0	99.82	0.08	80.62	19.33	0.82
1.0	1.0	0.1	99.85	0.07	84.55	15.38	0.56
1.0	1.0	0.5	99.84	0.08	79.74	20.20	0.58
1.0	1.0	1.0	99.81	0.10	71.37	28.58	0.53

Supplementary Table S2. More detailed performance of CompositeSearch. This table shows the true positive rate (TPR) and the false positive rate (FPR) of CompositeSearch when applied on two-components composites (Composite1) and three-components composites (Composite2). Identification is described as EXACT when the correct number of components is found and NON EXACT otherwise. The number of replicates is the same as in Supplementary Table S1. The FPR values represent occurrences of component sequences detected as composite.