Evolutionary Bioinformatics

SOFTWARE REVIEW

TIRfinder: A Web Tool for Mining Class II Transposons Carrying Terminal Inverted Repeats

Tomasz Gambin[1], Michał Startek[2], Krzysztof Walczak[1], Jarosław Paszek[3], Dariusz Grzebelus[4] and Anna Gambin[3,5]

[1]Institute of Computer Science, Warsaw University of Technology, Warsaw, Poland. [2]College of Inter-Faculty Individual Studies in Mathematics and Natural Sciences, University of Warsaw, Warsaw, Poland. [3]Institute of Informatics, University of Warsaw, Warsaw, Poland. [4]Department of Genetics, Plant Breeding and Seed Science, University of Agriculture in Krakow, Krakow, Poland. [5]Mossakowski Medical Research Centre Polish Academy of Sciences, Warsaw, Poland. Corresponding author email: tgambin@ii.pw.edu.pl

Abstract: Transposable elements (TEs) can be found in virtually all known genomes; plant genomes are exceptionally rich in this kind of dispersed repetitive sequences. Current knowledge on TE proliferation dynamics places them among the main forces of molecular evolution. Therefore efficient tools to analyze TE distribution in genomes are needed that would allow for comparative genomics studies and for studying TE dynamics in a genome. This was our main motivation underpinning TIRfinder construction—an efficient tool for mining class II TEs carrying terminal inverted repeats. TIRfinder takes as an input a genomic sequence and information on structural properties of a TE family, and identifies all TEs in the genome showing the desired structural characteristics. The efficiency and small memory requirements of our approach stem from the use of suffix trees to identify all DNA segments surrounded by user-specified terminal inverse repeats (TIR) and target site duplications (TSD) which together constitute a mask. On the other hand, the flexibility of the notion of the TIR/TSD mask makes it possible to use the tool for de novo detection. The main advantages of TIRfinder are its speed, accuracy and convenience of use for biologists. A web-based interface is freely available at http://bioputer.mimuw.edu.pl/tirfindertool/.

Keywords: transposon, terminal inverted repeat, suffix tree, comparative genomics

Evolutionary Bioinformatics 2013:9 17–27

doi: 10.4137/EBO.S10619

This article is available from http://www.la-press.com.

Introduction

Transposable elements (TEs) are DNA segments capable of changing their position in the genome. They are a major component of genomes of many organisms, comprising 66%–69% of the human genome[1] and even up to 85% of genetic material of certain plants (eg, maize).[2]

Recently, genomes of many species have been fully or partially sequenced and several bioinformatic tools have been created facilitating a more systematic identification and annotation of numerous TE families. On the basis of the mechanism of transposition, TEs were divided into two groups, ie, class I (retrotransposons) and class II (DNA transposons).[3,4] Retrotransposons transpose via an RNA intermediate (copy-and-paste mechanism), each transposition event leading to an increase of their copy number. DNA transposons change their chromosomal localization by physical excision of the element from the donor site and reintegration in the acceptor site. This group contains two subclasses, depending on the number of DNA strands cut during transposition. Subclass 1 is mobilized in accordance with the cut-and-paste mechanism, where both strands are cut at each end resulting in the complete excision of the TE. Most Subclass 1 DNA transposons carry Terminal Inverted Repeats (TIR) and create Target Site Duplication (TSD) upon insertion. Subclass 2 elements transpose on the basis of the rolling circle mechanism, where only one strand is cut. Classes and subclasses are further divided into orders, superfamilies, and families.

The effect of TE proliferation on genome evolution is noticeable.[5] Moreover, the importance of accurate TE annotation and masking for the structural characterization of newly sequenced genomes drives the interest in developing new methods for TE detection and analysis.[6] The exhaustive list of tools and resources for TE analysis compiled by Bergman Lab (http://bergmanlab.smith.man.ac.uk/) contains about 120 items. A large number of them are designed for particular TE families (eg, *Helitrons*,[7] LTR retrotransposons[8]) or for analysis of particular species (like *Drosophila melanogaster*[9]), several of these are for general use, ie, for all kinds of repeats (RepeatMasker,[10] REPuter,[11] PILER,[12] RepeatScout,[13] RECON[14]), and finally six of them are suitable for the structural analysis of class II elements (Inverted Repeat Finder,[15] MAK,[16] MITE-hunter,[17] MUST,[18] STAN,[19] TRANSPO[20]),with the latter three providing a web interface.

MUST[18] allows the user to search for all Miniature Inverted-repeat TEs (MITEs) that satisfy given criteria corresponding to minimum and maximum length of TIR, TSD and size of MITEs (up to 1000 bp). TRANSPO,[20] in addition to the functionality of MUST, enables the user to specify the sequence of TIRs and maximum number of errors allowed in TIRs. STAN[19] is the most flexible tool. It finds all sequences containing a given pattern specified in the SVG grammar. However, STAN uses fixed (non-parameterized) definition of inverted repeats which makes it less suitable for searching class II transposons.

All of the tools mentioned above have been developed to facilitate identification of MITEs, ie, elements that have TIRs but lack any coding capacity. Our TE discovery tool, TIRfinder, (which also captures specific structural features) offers functionality that goes beyond the proposed methods. It combines an efficient approach based on suffix trees that allows for de novo TE detection, with the possibility of a deep analysis of a specific TE family, based on its structural characteristics. In particular, while searching for all putative TEs, TIRfinder allows the user to specify TIR and TSD patterns as a sequence of A, C, T, G or symbols from extended IUPAC nomenclature.[21] This provides the ability to define TIR/TSD patterns as consensus sequences which combine conserved and non-conserved positions. A detailed description of the tool has not been provided previously, however the prototype, stand-alone version of TIRfinder was used for mining TEs in *Medicago truncatula*.[22,23]

Recently, deep sequencing projects have increased dramatically, raising the possibility of the repetitive DNA characterization of not completely sequenced organisms. Notice that our approach based on a suffix tree requires a large contiguous genomic sequence. Therefore for analysis of non-assembled raw reads produced by next generation sequencing, recently developed tools which utilize a clustering approach[24] should be applied.

Methods of TE Detection and Analysis

TIRfinder analysis workflow

TIRfinder allows efficient searches of the DNA for structured set of motifs that define TEs to be conducted. Then it performs a BLAST search to find transposase or any other TE-related open reading frames (ORFs), aiming at the detection of autonomous copies, as well as elements directly derived from them. The method works in three stages: structural analysis, functional analysis and MITE analysis (see Fig. 1).

In the first step, TIRfinder scans the input sequence for all putative TEs, based on the given structural characteristic of a particular TE family, ie, patterns describing TIR, TSD, the number of allowed mismatches and size limits of desired TEs. The algorithmic details of the structural analysis are described further in this section.

Next, all elements identified in the structural analysis stage are analyzed to check whether they contain the ORF coding for a protein specified by the user, most often it would be a transposase. For this purpose, they are aligned with the protein sequence (using an appropriate version of the BLAST algorithm)[25] and elements with statistically significant similarity are marked as putative autonomous. Each autonomous element is then used to search for so called Derivatives. These are non-autonomous elements that have emerged from an autonomous elements during the course of evolution, often by internal deletions disrupting the ORFs. To classify the TE as a derivative of a given autonomous element, we require that both pairs of subterminal regions share a significant level of similarity which is defined by the user (eg, BLAST E-value < e-10).

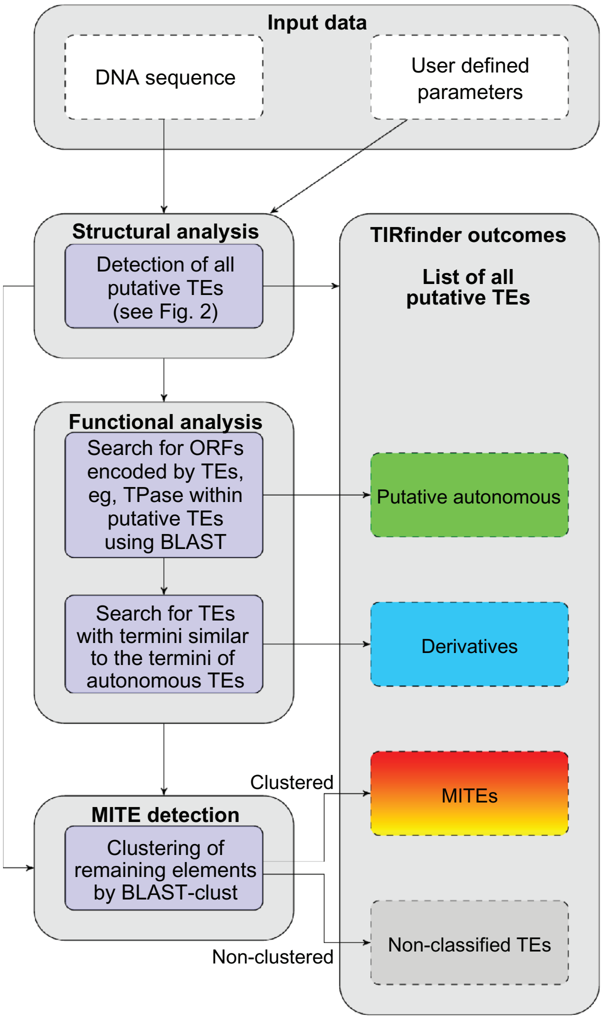

Input data
DNA sequence
User defined parameters
Structural analysis
Detection of all putative TEs (see Fig. 2)
TIRfinder outcomes
List of all putative TEs
Functional analysis
Search for ORFs encoded by TEs, eg, TPase within putative TEs using BLAST
Putative autonomous
Search for TEs with termini similar to the termini of autonomous TEs
Derivatives
Clustered
MITE detection
Clustering of remaining elements by BLAST-clust
MITEs
Non-clustered
Non-classified TEs

Figure 1. The control flow through different phases of the proposed TEs detection method.

Finally, the remaining set of putative TEs (ie, those not classified as autonomous TEs and their derivatives) is analyzed to find short elements which cluster together based on their sequence similarity. The level of similarity is defined by the user. Clustering is performed using BLAST-clust algorithm.[25] The TEs assigned to clusters are labeled as MITEs.

Structural analysis

Our application follows a structure-based approach, ie, it relies on the detection of specific models of TE architecture consisting of a pair of TIRs (inverted repeats) that are flanked by TSDs (direct repeats).

For detecting TEs we use suffix trees which are very efficient data structures, commonly used in computer science over last four decades.[26] They consists of a root, nodes and labeled edges representing one or more characters from input sequence. All suffixes of the input sequence can be obtained by traversing the suffix tree from the root to leaves. The core idea is that all suffixes that share the common prefix are hanged off on the common node, which reduces the total number of nodes and memory usage.

The algorithm implemented in TIRfinder takes as an input a DNA sequence, a mask corresponding to combined TSD and TIR patterns, and other parameters (ie, the number of mismatches), see Figure 2 for the pseudocode of the algorithm and Figure 3 for graphical explanation of the mask notion.

First, the DNA sequence is divided into a set of smaller fragments of the same length, corresponding to the maximal TE size. A suffix tree is built for every two consecutive fragments, with overlaps of one fragment length (see Fig. 3B). This is to ensure that we do not miss any match. The algorithm for each fragment independently searches all matches

```
Input:  DNA sequence $G$, TIR pattern, TSD pattern,
        max_size, mismatch_thresholds
Output:   all regions flanked by TIR's and TSD up to predefined
          mismatch threshold
        split the sequence $G$ into fragments $g_i$ $i = 1\ldots n$ of size
        = max_size
        MASK = TSD · TIR
        foreach sequence $g_i \cdot g_{i+1}$
            build the suffix tree $ST_i$;
            find all matches to MASK in $g_{i+1}$,
            say $m_1, m_2, \ldots m_j, \ldots m_k$;
            foreach $m_j$
                find in the suffix tree $ST_i$ all positions that
                match to revcomp($m_j$);
                // revcomp = reverse complement
                check the number of mismatches between
                MASK and revcomp($m_j$);
```

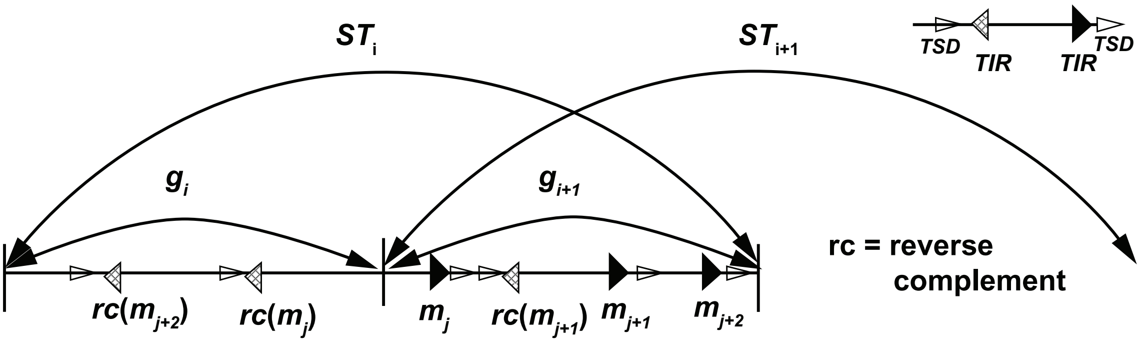


Figure 2. TIRfinder algorithm.

for the mask. In this step, all potential 3′-ends of a TIR should be found. Then we determine if the complementary (5′-end) part of the repeat exists, which is the most time-consuming part of the algorithm. In order to do this efficiently the suffix tree is used (it is built in linear time with respect to a length of the fragment).[27] For each match (TIR + TSD) found in the previous step, the reverse complement of TIR followed by TSD is searched in the suffix tree (see Fig. 2).

The principal advantage of our approach is memory efficiency: the genomic DNA sequence is split into smaller fragments and the suffix tree data structure needs only a linear amount of space. Thus, we do not impose any limits to the total length of the sequence and the stand-alone version of TIRfinder can be run even on standard PC computers.

Usage

The user must first select the genomic sequence to be searched for TEs. Several plant genomes are currently available in the TIRfinder website (*Arabidopsis thaliana, Arabidopsis lyrata, Oryza sativa japonica, Oryza sativa indica and Medicago truncatula*); other genomes can be provided upon request. Alternatively, the user can provide any sequence by uploading a FASTA file not exceeding 100 MB.

Next, the user has to define several parameters such as a minimal and maximal distance between TIRs and patterns of inverted and direct repeats. Patterns

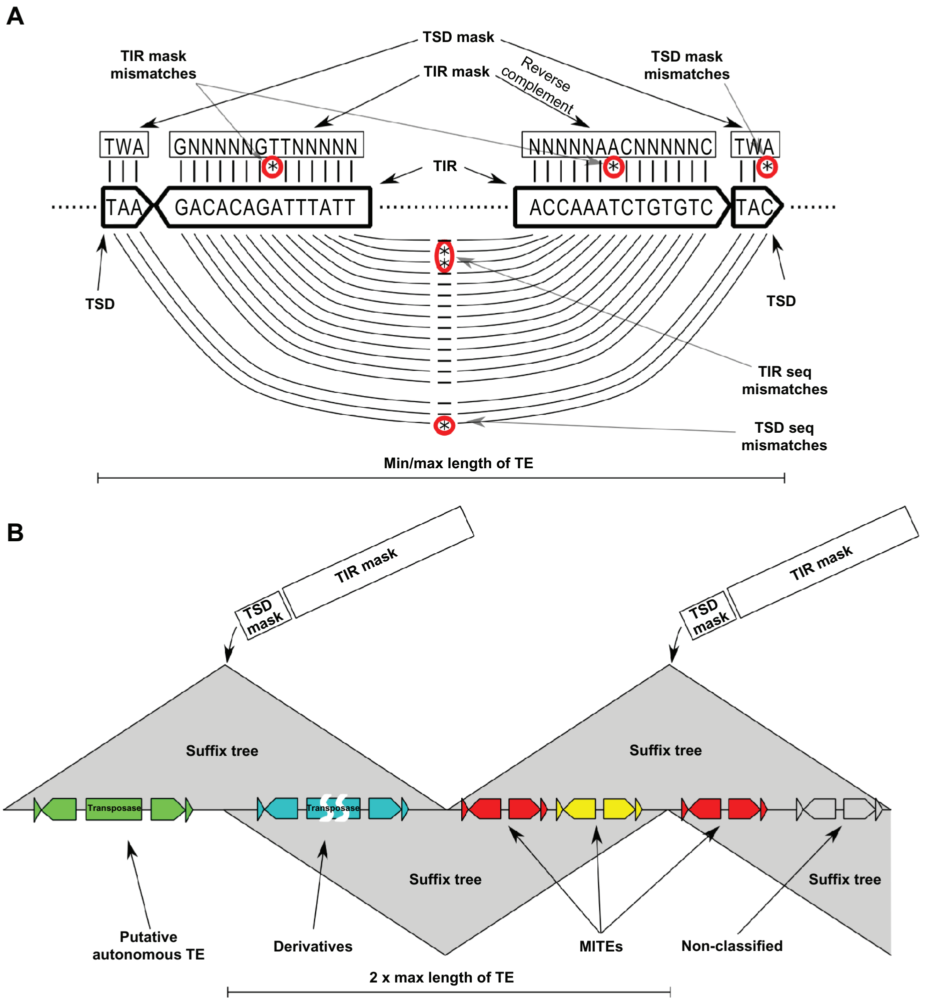


Figure 3. TIRfinder—structural analysis. (**A**) Explanation of TIR and TSD mask concept. (**B**) Overview of TEs detection phase.

of the particular TIR or TSD can be determined by means of a finite string composed of extended IUPAC nucleotide alphabet.[21]

Note that the user has to define only one side of the repeat. The second part will be computed as a reverse complement in the case of TIR or simply duplicated in the case of TSD. Furthermore, there is a possibility to specify a maximal number of mismatches between TSDs and TIRs flanking each copy of identified TEs. TIRfinder reads a DNA sequence given in a FASTA file to search for results and saves them in a simple text format (see Appendix for the detailed description). Then the file can be further processed in order to obtain more detailed and accurate data.

Carefully prepared input data, such as TIR and TSD masks, are extremely important in order to get satisfactory results. One possible way is to take TIR sequences from individual copies of TEs representing the desired family and their corresponding flanking TSDs, and create consensus for TSD and TIR sequences. Afterwards, the user can set up other parameters of TIRfinder, which allow for the control of similarity between mined elements and the mask (MaskMismatches) or between corresponding 5′- and 3′- TIRs and TSDs of found elements (SeqMismatches) see Figure 3A. The ability to manipulate these parameters makes it easier to search for known, well conserved elements as well as to mine new (sub)families of TEs de novo.

Subsequently, the user may specify parameters for the identification of TE copies carrying ORFs, which for simplicity was dubbed functional analysis. First, the protein sequence (in FASTA format) is provided to detect ORFs (a significance threshold is required). Then, long non-autonomous copies—direct derivatives are identified as elements sharing similarity of subterminal regions (length parameter in bp and

Table 1. Pogo-like TEs found by TIRfinder vs. annotated in Repbase.

i[a]	chr1	chr2	chr3	chr4	chr5	Sum	Positive predictive value[b]
0	1	2	0	4	2	9	100
1	11	6	7	10	9	43	100
2	20	10	16	17	13	76	95
3	25	20	18	20	14	97	94
4	29	22	22	22	17	112	88
5	33	22	25	24	20	124	84
Rep.[c]	25	22	20	22	8	97	

Notes: [a]Number of allowed TIR mask and TIR seq mismatches; [b]% of TIRfinder output masked by Repbase data. [c]numbers of *ATHPOGO* elements (>300 bp) annotated in Repbase.

alignment significance threshold are provided by the user) with one of the previously found autonomous TEs. If the ORF sequence is unknown to the user, the functional analysis step can be omitted in the course of analysis.

Finally, the user defines constraints for MITE analysis, ie,: minimal and maximal length of MITEs, maximal number of MITE clusters and the level of in-cluster similarity.

TIRfinder implementation

Current release of TIRfinder is an open source web application built with Java, Perl, Apache and other tools (ie, BLAST, BLAST-clust). Previous, standalone release of TIRfinder, used for case studies reported in,[22,23] is available at http://sourceforge.net/projects/tirfinder/.

The program is organized into web tier and background processes. Web tier includes a set of Perl scripts responsible for communication with user, execution of subsequent tasks, and results presentation. Time consuming operations (ie, structural analysis, BLAST alignment, clustering) are performed as background processes, which prevents the user interface from freezing during calculations.

The application is hosted on one of the University of Warsaw servers with 16-cores and 64GB of RAM.

Case Studies

As an example of TIRfinder application, we searched the genome of *Arabidopsis thaliana* and *Medicago Truncatula* for the *ATHPOGON3* class II TE family.[28] We fixed TIR and TSD strings as consensus subsequences from given sequences of *ATHPOGO*, *ATHPOGON1*, *ATHPOGON2* and *ATHPOGON3* from Repbase.[29] Finally, we obtained the following parameters: TIR pattern = CAGTARAAMCTCTATAAATTAATA, TSD pattern = TA. We decided to set min distance = 300, max distance = 5000, max TSD mask and TSD seq mismatches = 0, max TIR mask and TIR seq mismatches = $i \in (0, 1, 2, 3, 4, 5)$. The experiment allowed for efficient mining of pogo-like transposons in *A. thaliana* and *M. truncatula*. The number of TEs found by TIRfinder and annotated in Repbase for each chromosome of *A. thaliana* is shown in the Table 1 (see also Appendix). The breakdown analysis of the found TE sizes (shown in Fig. 4) reveals that lengths of the majority of putative TEs correspond to the sizes of known pogo-like elements. Most of these elements were MITEs, while in *M. truncatula* we found only 28 pogo-like elements and no MITEs. It fully confirmed previous reports on these elements, reflecting species-specific behavior of related TEs in the two species[30] and confirming TIRfinder efficiency.

Moreover, we performed similar search of *M. truncatula* genome for *PIF/Harbinger* TEs family (using TIR pattern = GNNNNNGTTNNNNN and TSD pattern = TWA). The exemplary results of this analysis, presented by TIRfinder (see Fig. 5),

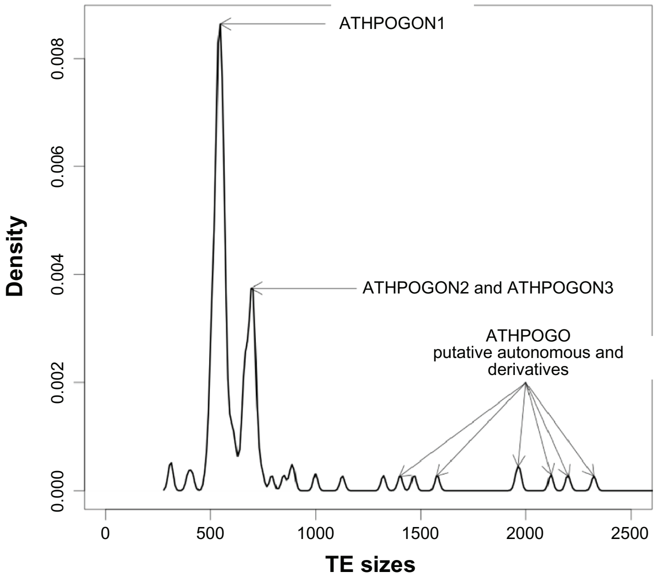
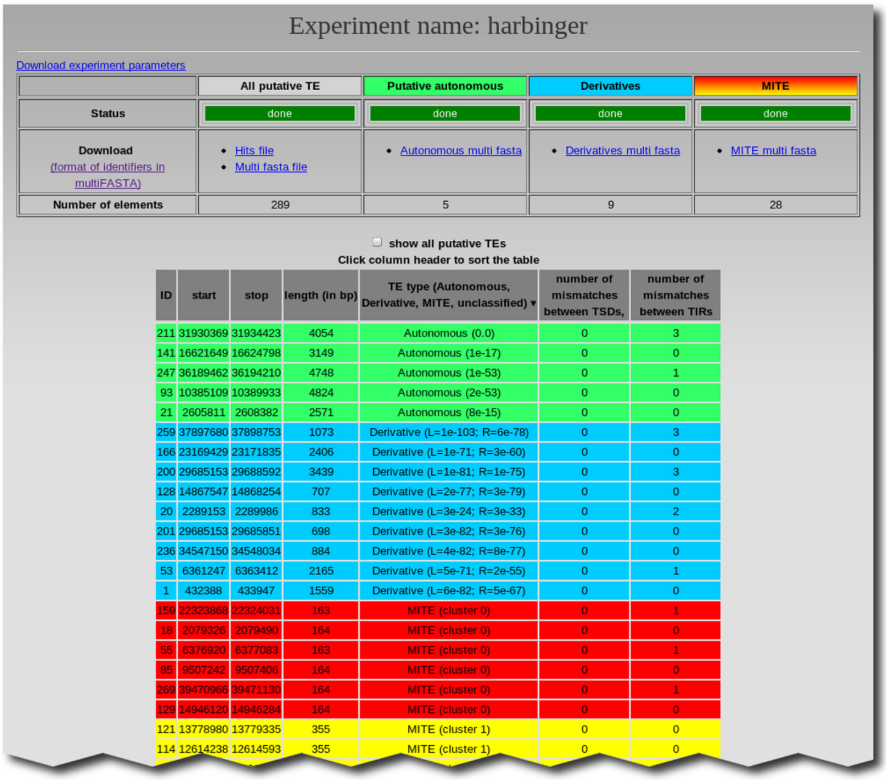


Figure 4. Pogo-like TEs landscape detected by TIRfinder in *A. thaliana*.



Experiment name: harbinger

Download experiment parameters

	All putative TE	Putative autonomous	Derivatives	MITE
Status	done	done	done	done
Download (format of identifiers in multiFASTA)	• Hits file • Multi fasta file	• Autonomous multi fasta	• Derivatives multi fasta	• MITE multi fasta
Number of elements	289	5	9	28

□ show all putative TEs

Click column header to sort the table

ID	start	stop	length (in bp)	TE type (Autonomous, Derivative, MITE, unclassified) ▾	number of mismatches between TSDs,	number of mismatches between TIRs
211	31930369	31934423	4054	Autonomous (0.0)	0	3
141	16621649	16624798	3149	Autonomous (1e-17)	0	0
247	36189462	36194210	4748	Autonomous (1e-53)	0	1
93	10385109	10389933	4824	Autonomous (2e-53)	0	0
21	2605811	2608382	2571	Autonomous (8e-15)	0	0
259	37897680	37898753	1073	Derivative (L=1e-103; R=6e-78)	0	3
166	23169429	23171835	2406	Derivative (L=1e-71; R=3e-60)	0	0
200	29685153	29688592	3439	Derivative (L=1e-81; R=1e-75)	0	3
128	14867547	14868254	707	Derivative (L=2e-77; R=3e-79)	0	0
20	2289153	2289986	833	Derivative (L=3e-24; R=3e-33)	0	2
201	29685153	29685851	698	Derivative (L=3e-82; R=3e-76)	0	0
236	34547150	34548034	884	Derivative (L=4e-82; R=8e-77)	0	0
53	6361247	6363412	2165	Derivative (L=5e-71; R=2e-55)	0	1
1	432388	433947	1559	Derivative (L=6e-82; R=5e-67)	0	0
159	22323868	22324031	163	MITE (cluster 0)	0	1
18	2079326	2079490	164	MITE (cluster 0)	0	0
55	6376920	6377083	163	MITE (cluster 0)	0	1
85	9507242	9507406	164	MITE (cluster 0)	0	0
269	39470966	39471130	164	MITE (cluster 0)	0	1
129	14946120	14946284	164	MITE (cluster 0)	0	0
121	13778980	13779335	355	MITE (cluster 1)	0	0
114	12614238	12614593	355	MITE (cluster 1)	0	0

Figure 5. The example of TIRfinder outcomes: search for *PIF*/*Harbinger* TEs family in chromosome 5 of *M. truncatula* genome.

reveals the occurrence of all functional classes of *PIF/Harbinger* TEs, ie, putative autonomous, derivatives and MITEs.

Discussion

Efficient methods for computational analysis of TEs are of great interest in the light of the TE contribution to genome structure and evolution. We developed TIRfinder to provide users with a fully functional web server that allows structure-based detection and clustering of class II TEs carrying TIRs.

There are several unique features of TIRfinder, in comparison with other tools designed for mining TEs carrying TIRs. It provides exibility in defining TIR/TSD masks, allowing for a range of actions, from de novo identification of TEs belonging to a given superfamily in newly annotated genomes[22] to extensive mining of all copies belonging to a particular family (eg, *ATHPOGO*), depending on the mask specificity.

Although our tool is highly efficient for well-determined TSD/TIR mask, the complexity of the method may significantly increase with relaxing the mask constraints (eg, allowing large number of mismatches between TIR/TSD). Moreover, TIRfinder helps identifying elements with coding capacity, however some additional downstream processing, eg, ORF prediction, should be performed for all copies classified as autonomous, if a more detailed functional characteristics is required.

Author Contributions

TG, MS, JP developed the software and performed tests. AG, KW and DG supervised all the software developments, and carried case studies. TG, MS, DG, and AG wrote the manuscript. All authors reviewed and approved of the final manuscript.

Funding

Author(s) disclose no funding sources.

Acknowledgements

This work was partially supported by Polish Ministry of Science and Educations grant N301 013436, and European Social Fund (UDA-POKL.04.01.01-00-072/09-00 and UDA-POKL.04.01.01-00-051/10-00).

Competing Interests

Author(s) disclose no potential conflicts of interest.

Disclosures and Ethics

As a requirement of publication author(s) have provided to the publisher signed confirmation of compliance with legal and ethical obligations including but not limited to the following: authorship and contributorship, conflicts of interest, privacy and confidentiality and (where applicable) protection of human and animal research subjects. The authors have read and confirmed their agreement with the ICMJE authorship and conflict of interest criteria. The authors have also confirmed that this article is unique and not under consideration or published in any other publication, and that they have permission from rights holders to reproduce any copyrighted material. Any disclosures are made in this section. The external blind peer reviewers report no conflicts of interest.

References

1. de Koning AP, Gu W, Castoe TA, Batzer MA, Pollock DD. Repetitive elements may comprise over two-thirds of the human genome. *PLoS Genet*. Dec 2011;7(12):e1002384. Epub Dec 1, 2011.
2. Schnable PS, Ware D, Fulton RS, et al. The B73 maize genome: complexity, diversity, and dynamics. *Science*. Nov 20, 2009;326(5956):1112–5.
3. Kapitonov VV, Jurka J. A universal classification of eukaryotic transposable elements implemented in Repbase. *Nat Rev Genet*. May 2008;9(5):411–2; author reply 414.
4. Wicker T, Sabot F, Hua-Van A, et al. A unified classification system for eukaryotic transposable elements. *Nat Rev Genet*. Dec 2007;8(12): 973–82.
5. Britten RJ. Transposable element insertions have strongly affected human evolution. *Proc Natl Acad Sci U S A*. Nov 16, 2010;107(46):19945–8. Epub Nov 1, 2010.
6. Bergman CM, Quesneville H. Discovering and detecting transposable elementsin genome sequences. *Brief Bioinform*. Nov 2007;8(6):382–92. Epub Oct 10, 2007.
7. Du C, Caronna J, He L, Dooner H. Computational prediction and molecular confirmation of Helitron transposons in the maize genome. *BMC Genomics*. 2008;9:51.
8. Ellinghaus D, Kurtz S, Willhoeft U. LTRharvest, an efficient and flexible software for de novo detection of LTR retrotransposons. *BMC Bioinformatics*. 2008;9:18.
11. Kurtz S, Schleiermacher C. REPuter: fast computation of maximal repeats in complete genomes. *Bioinformatics*. May 1999;15(5):426–7.
14. Bao Z, Eddy SR. Automated de novo identification of Repeat Sequence families in sequenced genomes. *Genome Res*. 2002;12(8):1269–76.
15. Warburton PE, Giordano J, Cheung F, Gelfand Y, Benson G. Inverted repeat structure of the human genome: the X-chromosome contains a preponderance of large, highly homologous inverted repeats that contain testes genes. *Genome Res*. Oct 2004;14(10A):1861–9.
16. Yang G, Hall TC. MAK, a computational tool kit for automated MITE analysis. *Nucleic Acids Res*. Jul 1, 2003;31(13):3659–65.
17. Han Y, Wessler SR. MITE-Hunter: a program for discovering miniature inverted-repeat transposable elements from genomic sequences. *Nucleic Acids Res*. Dec 2010;38(22):e199. Epub Sep 29, 2010.
18. Chen Y, Zhou F, Li G, Xu Y. MUST: a system for identification of miniature inverted-repeat transposable elements and applications to Anabaena variabilis and Haloquadratum walsbyi. *Gene*. May 1, 2009;436(1–2):1–7. Epub Feb 10, 2009.
19. Nicolas J, Durand P, Ranchy G, Tempel S, Valin AS. Suffix-tree analyser (STAN): looking for nucleotidic and peptidic patterns in chromosomes. *Bioinformatics*. Dec 15, 2005;21(24):4408–10. Epub Oct 13, 2005.
20. Santiago N, Herráiz C, Goñi JR, Messeguer X, Casacuberta JM. Genome-wide analysis of the Emigrant family of MITEs of Arabidopsis thaliana. *Mol Biol Evol*. Dec 2002;19(12):2285–93.
21. Cornish-Bowden A. Nomenclature for incompletely specified bases in nucleic acid sequences: recommendations 1984. *Nucleic Acids Res*. May 10, 1985;13(9):3021–30.
22. Grzebelus D, Lasota S, Gambin T, Kucherov G, Gambin A. Diversity and structure of PIF/Harbinger-like elements in the genome of Medicago truncatula. *BMC Genomics*. Nov 9, 2007;8:409.
23. Grzebelus D, Gładysz M, Macko-Podgórni A, et al. Population dynamics of miniature inverted-repeat transposable elements (MITEs) in Medicago truncatula. *Gene*. Dec 15, 2009;448(2):214–20. Epub Jun 17, 2009.
24. Novák P, Neumann P, Macas J. Graph-based clustering and characterization of repetitive sequences in next-generation sequencing data. *BMC Bioinformatics*. Jul 15, 2010;11:378.
25. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ. Basic local alignment search tool. *J Mol Biol*. Oct 5, 1990;215(3):403–10.
28. Le QH, Wright S, Yu Z, Bureau T. Transposon diversity in Arabidopsis thaliana. *Proc Natl Acad Sci U S A*. Jun 20, 2000;97(13):7376–81.
29. Jurka J, Kapitonov VV, Pavlicek A, Klonowski P, Kohany O, Walichiewicz J. Repbase Update, a database of eukaryotic repetitive elements. *Cytogenet Genome Res*. 2005;110(1–4):462–7.
30. Guermonprez H, Loot C, Casacuberta JM. Different strategies to persist: the pogo-like Lemi1 transposon produces miniature inverted-repeat transposable elements or typical defective elements in different plant genomes. *Genetics*. Sep 2008;180(1):83–92. Epub Aug 30, 2008.

Appendix

Output file formats

TE identifiers

All groups of detected TE elements, ie, all putative TEs, putative autonomous TEs, derivatives and MITEs can be downloaded as separate multi fasta files. Names of each TE sequence in the file is composed as in the following example:

For the name “1-102-101894195-101894898-ATHPOGO”

1 = chromosome number
102 = ID of putative TE
101894195 = TE start position
101894898 = TE end position
ATHPOGO = TE family name

Hits file

The output of structural analysis (containing all putative TEs) can be downloaded as txt file. First 5 lines consists of TIR/TSD mask parameters used to search for TEs. Next the total number of found TEs is presented. Subsequent lines describes each found TE including:

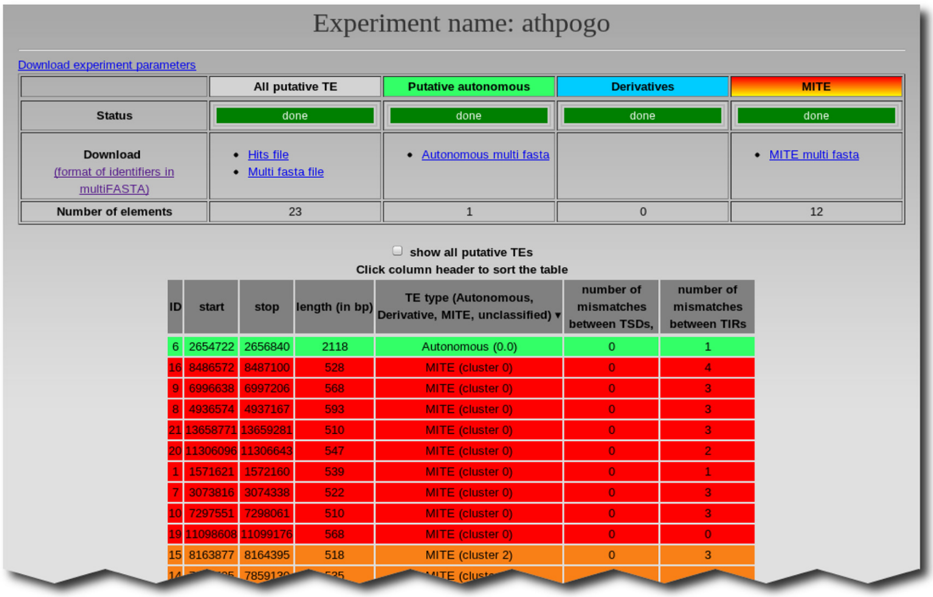
- element ID
- start and stop genomic position
- TSD/TIR sequence and mask mismatches
- TSD and TIR sequence alignment
- Extracted sequence

The begining of the hits file corresponding to the results shown in Figure A2 is presented below:

```
MASK ID = 0
MASK(TSD:TIR) = TA:CAGTARAAMCTCTATAAATTAATA
max mask mismatches(tsd:tir) = 0:5
max seq mismatches(tsd:tir) = 0:5
distance(min:max) = 200:5000
------------------------------------------------------------------------
Number of found elements = 126
------------------------------------------------------------------------
ID: 0 begin: 784927 end: 785436 length: 509
mismatches seq1 to mask(tsd:tir) = 0 : 0
mismatches seq2 to mask(tsd:tir) = 0 : 2
mismatches between seq1 and seq2(tsd:tir:extra) = 0 : 2 : 62
TA:CAGTAAAACCTCTATAAATTAATA:GTATTGAGACCAAAAAAATTGATTAATTTAGAGAG ...
||:|||||||| || ||||||||||||: |    | |  || ||  ||  || ||||  | |    ...
TA:CAGTAAAATCTTTATAAATTAATA:TTCGATAAATTAATAATTTTTATAAATTAATAATT ...

***************SEQ START*************

TACAGTAAAACCTCTATAAATTAATAGTATTGAGACCAAAAAAATTGATTAATTTAGAGA
GATATTAATTTATCGATAAACTAATAAAATTGTACAAATTTGTAAATTCTTACCAAAATT
CTATAAAAAGTGTAGTTTTTTCTTTATAATCAATAGTAATTACAAGTAAAAAGCAATTAT
TAAATTTAAAAAACAATACAATTTTTTTTTATGTTGTAAAATACTAGAATTGAACACTAT
AAAAAATTAATAAAAATTGAAGAAAACAATTAGTACAGCCATATTTTACATATGATATTT
```

```
ATAATATATATTAGTAAATTTATAAAATTATTAATTTATACTATTGATGGGAGCATATAT
TTATATAAGATTTTCAATAAAAATATTATCTTATTAGTTTATCGATTTATATCAATTTTT
ATATTGGTCTCAACTCTGCACCAAAAAAAATTATTAATTTATAAAAATTATTAATTTATC
GAATATTAATTTATAAAGATTTTACTGTA

***************SEQ END***************

------------------------------------------------------------------------
ID: 1 begin: 6759166 end: 6759752 length: 586
mismatches seq1 to mask(tsd:tir) = 0 : 0
mismatches seq2 to mask(tsd:tir) = 0 : 2
mismatches between seq1 and seq2(tsd:tir:extra) = 0 : 2 : 59
TA:CAGTAAAACCTCTATAAATTAATA:ATATTTGGACCAATAAAATTTATTAATTTAGAGAG ...
||:||| ||||| ||||||||||||||: | | |||||| ||| | | | | ...
TA:CAGCAAAACTTCTATAAATTAATA:CTTGAAAAATTAATAAATTTTTCAAGTATTAATTT ...

***************SEQ START**************

TACAGTAAAACCTCTATAAATTAATAATATTTGGACCAATAAAATTTATTAATTTAGAGA
GGTATTAATTTATCGATAAATAAATAAAATTGTACAAATTTTGTAAATTCTTACAAAAAT
TCGATATAAAATATAGTTTTTTTCCCTATAATCAATAGTATTTCTAAAACCAATTACAAG
TAGAAAGCAATTATTAAGTTTGAAAAACAATACAATTTTTTCGATATTGTAAAATATTAG
AATTGAACACTACAAAGAATTAACAAAAATTGAAGAAAAACATTACTAGACTCATATTTT
ACATATGATATTTATAATATATATTGATAAATTTATATAATTATTAATTTATACTATTGA
TGGACCATATAATTATATAAGATTTTCAAAAATATTATTATCTTATTAATTTATCGATTT
TTATCAATTTTTACATTGGTCCCAACTTGCTCAGAATCTGTGCTCATTCTTATGCATTCT
AAAGAACTTGCTCAGACTCTCTGCTCATTACAGCAAAACTTCTATAAATTAATACTTGAA
AAATTTATTAATTTTTCAAGTATTAATTTATAGAAGTTTTGCTGTA

***************SEQ END***************
```

ATHPOGO TEs in *A. thaliana* genome



Experiment name: athpogo

Download experiment parameters

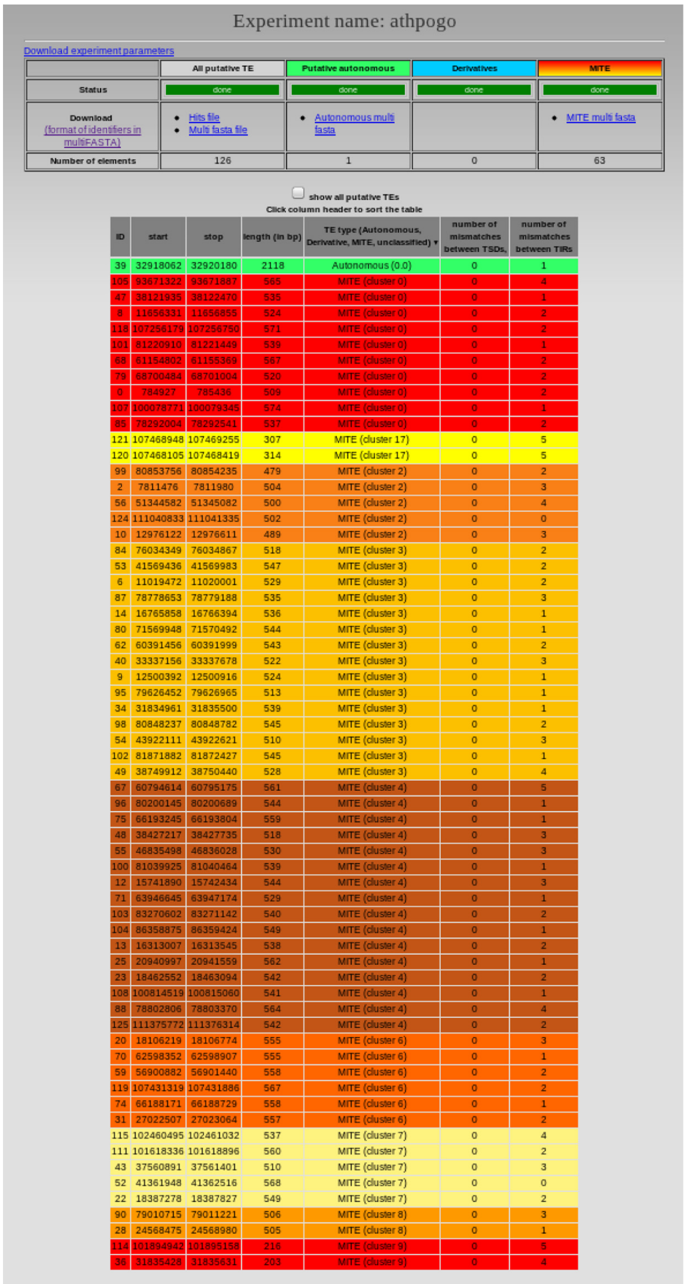
	All putative TE	Putative autonomous	Derivatives	MITE
Status	done	done	done	done
Download (format of identifiers in multiFASTA)	Hits file Multi fasta file	Autonomous multi fasta		MITE multi fasta
Number of elements	23	1	0	12

□ show all putative TEs

Click column header to sort the table

ID	start	stop	length (in bp)	TE type (Autonomous, Derivative, MITE, unclassified) ▾	number of mismatches between TSDs,	number of mismatches between TIRs
6	2654722	2656840	2118	Autonomous (0.0)	0	1
16	8486572	8487100	528	MITE (cluster 0)	0	4
9	6996638	6997206	568	MITE (cluster 0)	0	3
8	4936574	4937167	593	MITE (cluster 0)	0	3
21	13658771	13659281	510	MITE (cluster 0)	0	3
20	11306096	11306643	547	MITE (cluster 0)	0	2
1	1571621	1572160	539	MITE (cluster 0)	0	1
7	3073816	3074338	522	MITE (cluster 0)	0	3
10	7297551	7298061	510	MITE (cluster 0)	0	3
19	11098608	11099176	568	MITE (cluster 0)	0	0
15	8163877	8164395	518	MITE (cluster 2)	0	3
14	[illegible]	7859130	525	MITE (clust		

Figure A1. TIRfinder results of search for pogo-like family in chromosome 2 of *A. thaliana* genome.



Experiment name: athpogo

Download experiment parameters

	All putative TE	Putative autonomous	Derivatives	MITE
Status	done	done	done	done
Download (format of identifiers in multiFASTA)	Hits file Multi fasta file	Autonomous multi fasta		MITE multi fasta
Number of elements	126	1	0	63

□ show all putative TEs

Click column header to sort the table

ID	start	stop	length (in bp)	TE type (Autonomous, Derivative, MITE, unclassified) ▾	number of mismatches between TSDs,	number of mismatches between TIRs
39	32918062	32920180	2118	Autonomous (0.0)	0	1
105	93671322	93671887	565	MITE (cluster 0)	0	4
47	38121935	38122470	535	MITE (cluster 0)	0	1
8	11656331	11656855	524	MITE (cluster 0)	0	2
118	107256179	107256750	571	MITE (cluster 0)	0	2
101	81220910	81221449	539	MITE (cluster 0)	0	1
68	61154802	61155369	567	MITE (cluster 0)	0	2
79	68700484	68701004	520	MITE (cluster 0)	0	2
0	784927	785436	509	MITE (cluster 0)	0	2
107	100078771	100079345	574	MITE (cluster 0)	0	1
85	78292004	78292541	537	MITE (cluster 0)	0	2
121	107468948	107469255	307	MITE (cluster 17)	0	5
120	107468105	107468419	314	MITE (cluster 17)	0	5
99	80853756	80854235	479	MITE (cluster 2)	0	2
2	7811476	7811980	504	MITE (cluster 2)	0	3
56	51344582	51345082	500	MITE (cluster 2)	0	4
124	111040833	111041335	502	MITE (cluster 2)	0	0
10	12976122	12976611	489	MITE (cluster 2)	0	3
84	76034349	76034867	518	MITE (cluster 3)	0	2
53	41569436	41569983	547	MITE (cluster 3)	0	2
6	11019472	11020001	529	MITE (cluster 3)	0	2
87	78778653	78779188	535	MITE (cluster 3)	0	3
14	16765858	16766394	536	MITE (cluster 3)	0	1
80	71569948	71570492	544	MITE (cluster 3)	0	1
62	60391456	60391999	543	MITE (cluster 3)	0	2
40	33337156	33337678	522	MITE (cluster 3)	0	3
9	12500392	12500916	524	MITE (cluster 3)	0	1
95	79626452	79626965	513	MITE (cluster 3)	0	1
34	31834961	31835500	539	MITE (cluster 3)	0	1
98	80848237	80848782	545	MITE (cluster 3)	0	2
54	43922111	43922621	510	MITE (cluster 3)	0	3
102	81871882	81872427	545	MITE (cluster 3)	0	1
49	38749912	38750440	528	MITE (cluster 3)	0	4
67	60794614	60795175	561	MITE (cluster 4)	0	5
96	80200145	80200689	544	MITE (cluster 4)	0	1
75	66193245	66193804	559	MITE (cluster 4)	0	1
48	38427217	38427735	518	MITE (cluster 4)	0	3
55	46835498	46836028	530	MITE (cluster 4)	0	3
100	81039925	81040464	539	MITE (cluster 4)	0	1
12	15741890	15742434	544	MITE (cluster 4)	0	3
71	63946645	63947174	529	MITE (cluster 4)	0	1
103	83270602	83271142	540	MITE (cluster 4)	0	2
104	86358875	86359424	549	MITE (cluster 4)	0	1
13	16313007	16313545	538	MITE (cluster 4)	0	2
25	20940997	20941559	562	MITE (cluster 4)	0	1
23	18462552	18463094	542	MITE (cluster 4)	0	2
108	100814519	100815060	541	MITE (cluster 4)	0	1
88	78802806	78803370	564	MITE (cluster 4)	0	4
125	111375772	111376314	542	MITE (cluster 4)	0	2
20	18106219	18106774	555	MITE (cluster 6)	0	3
70	62598352	62598907	555	MITE (cluster 6)	0	1
59	56900882	56901440	558	MITE (cluster 6)	0	2
119	107431319	107431886	567	MITE (cluster 6)	0	2
74	66188171	66188729	558	MITE (cluster 6)	0	1
31	27022507	27023064	557	MITE (cluster 6)	0	2
115	102460495	102461032	537	MITE (cluster 7)	0	4
111	101618336	101618896	560	MITE (cluster 7)	0	2
43	37560891	37561401	510	MITE (cluster 7)	0	3
52	41361948	41362516	568	MITE (cluster 7)	0	0
22	18387278	18387827	549	MITE (cluster 7)	0	2
90	79010715	79011221	506	MITE (cluster 8)	0	3
28	24568475	24568980	505	MITE (cluster 8)	0	1
114	101894942	101895158	216	MITE (cluster 9)	0	5
36	31835428	31835631	203	MITE (cluster 9)	0	4

Figure A2. TIRfinder results of search for pogo-like family in the whole *A. thaliana* genome.

Notes: The single known autonomous TE was detected (see the green row in the table) as well as 125 other putative TEs. The 63 of them were classified as MITEs.