Desarrollo de una herramienta computacional actualizada para la identificación de Tandem Repeats en secuencias de ADN

María Paula Franco Guzmán
Universidad de los Andes
E-mail: mp.franco10@uniandes.edu.co
Bogotá, Colombia

17 de diciembre de 2020

1. Introducción

ADN es el nombre químico de la molécula que contiene la información genética en todos los seres vivos. El ADN se compone de dos cadenas que se enrollan entre ellas, formando una estructura de doble hélice. Cada cadena se compone de una parte central compuesta de azúcares y grupos fosfato. Anclado a cada azúcar, se encuentra una de 4 bases: adenina (A), citosina (C), guanina (G), y timina (T). Los segmentos cortos de ADN constituyen los genes, que son la base de la herencia. Estos contienen información de cómo se construyen proteínas, que estructuran órganos y tejidos del cuerpo humano. A su vez, estas proteínas se encargan de mover moléculas entre células, y generan reacciones químicas en el cuerpo.
Las moléculas de ADN están sujetas a una gran variedad de mutaciones debido a cambios en su estructura. Estas mutaciones en la estructura pueden provocar que las instrucciones para fabricar las proteínas encargadas de distintos procesos se modifiquen, llevando a un funcionamiento incorrecto de procesos bioquímicos por proteínas defectuosas o faltantes, y derivando así en afecciones de salud como lo son las enfermedades genéticas.

Uno de estos fenómenos mutacionales es el de los tandem repeats, o repeticiones en tándem. En este caso, un tramo de la secuencia de ADN se repite a través de esta. Estas repeticiones se encuentran adyacentes entre sí. Sin embargo, muchas veces se presentan pequeñas alteraciones en estas repeticiones, como pequeñas inserciones o eliminaciones. Un ejemplo de secuencia repetida podría ser la AT, de esta manera el tandem repeat se vería como $ATATATAT$.

Los tandem repeats (TRs) pueden ser definidos de acuerdo a varias características, como lo son el tamaño de unidad (medida del patrón que se repite), el número de copias de ese patrón, el tamaño total que ocupa el TR en el genoma, la ubicación del TR en el genoma, así como las impurezas que este posea. En la figura 1 se pueden observar estas características. Se puede observar el número de unidades en la repetición, es decir, cuántas veces se repite una secuencia en la cadena. Se puede observar el concepto de pureza, teniendo en cuenta que las repeticiones pueden tener pequeñas modificaciones. Por último, se puede observar el largo de la unidad, que indica cuántas bases componen la secuencia repetida.

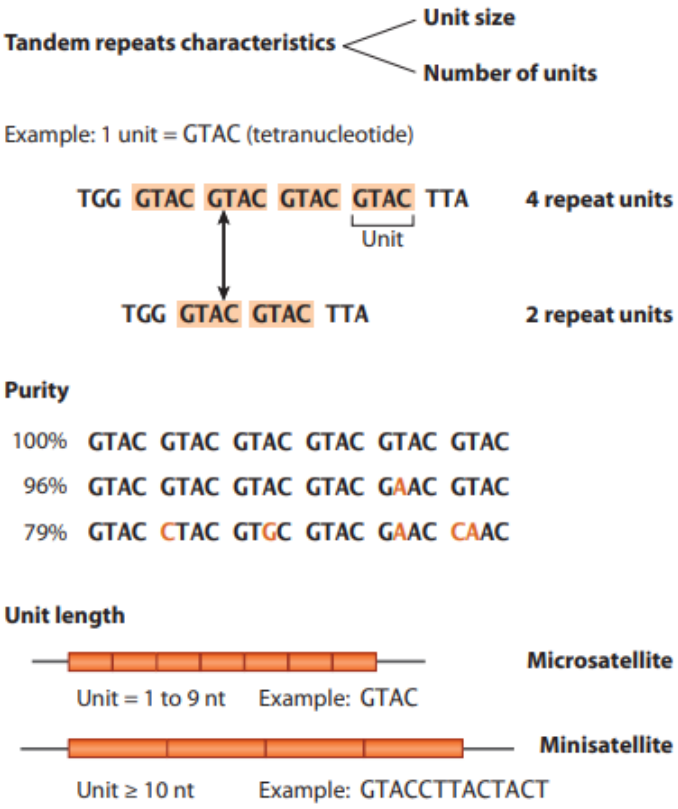

Figura 1: Características de los TR, tomado de [11]

Dependiendo del número de bases que compongan al TR se definen 3 tipos de TR, que son microsatélites, minisatélites y macrosatélites. En la figura 2 se puede observar el tamaño de cada tipo de TR.

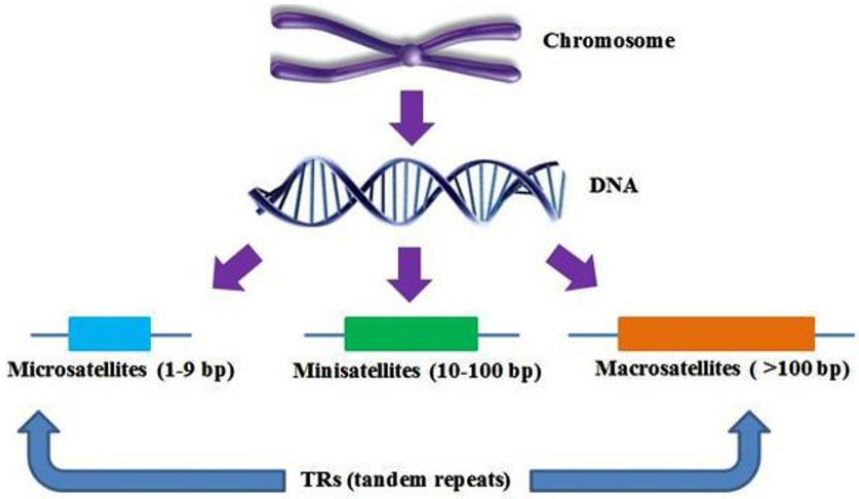

Figura 2: Tipos de TR, tomado de [13]

Es sabido que los tandem repeats ocurren frecuentemente en secuencias genómicas, comprendiendo al menos el 10 % del genoma humano. Los TR (Tandem Repeats) pueden estar involucrados en múltiples procesos celulares, incluyendo la expresión de los genes. Las mutaciones en estas secuencias repetidas pueden resultar en consecuencias fenotípicas, lo cual puede llevar a la presencia de enfermedades genéticas, como por ejemplo la enfermedad de Huntington, que ocasiona el desgaste de las neuronas del cerebro, atrofias musculares, deterioro en el equilibrio y coordinación (ataxia), entre otras afecciones. La referencia [11] puede ser consultada para más detalle de las implicaciones de los TR.

Teniendo en cuenta las grandes implicaciones que conllevan la presencia y mutación de los TR en el genoma humano, la comunidad bio-informática ha hecho múltiples esfuerzos por desarrollar proyectos de investigación que permitan realizar análisis a larga escala de estas repeticiones en secuencias de ADN. Así entonces, dado un genoma de referencia surge un problema clásico de bioinformática, que consiste en identificar los TR en un ensamblaje de genoma. Sin embargo, encontrar y validar TRs puede llegar a ser una tarea muy complicada, debido a que el éxito de estos proyectos depende enteramente de la sensibilidad e implementación de los algoritmos que se utilicen.

2. Marco teórico

- **ADN** El ácido desoxirribonucleico es el compuesto químico que contiene las instrucciones necesarias para desarrollar y dirigir las actividades de los organismos vivientes. Las moléculas de ADN están hechas de un par de cadenas entrelazadas.
  Cada cadena de ADN está hecha de 4 unidades químicas, llamadas bases nitrogenadas o nucleobases, que abarcan el alfabeto genético. Estas bases son adenina (A), timina (T), guanina (G) y citosina (C). Las bases en cadenas opuestas se complementan en pares de la siguiente manera: a una A siempre le corresponde una T, a una C siempre le corresponde una G. El orden de las A, T, C y G determina el significado de la información codificada en esa parte de la molécula del ADN, así como el orden de las letras en una palabra determina su significado.

- **Genoma** El conjunto completo de ADN de un organismo es llamado genoma. Cada célula en el cuerpo contiene una copia completa de aproximadamente 3 mil millones de pares de bases, o letras, que constituyen el genoma humano.
  Con este lenguaje de cuatro letras, el ADN contiene la información necesaria para construir enteramente el cuerpo humano. Un gen tradicionalmente hace referencia a la unidad de ADN que contiene instrucciones para construir una proteína o un conjunto de proteínas específico. Estas proteínas forman parte de estructuras del cuerpo tales como órganos y tejidos, a su vez que controlan reacciones químicas y llevan señales entre células. Si el ADN de una célula muta, una proteína anormal puede ser producida, lo que puede interrumpir los procesos usuales del cuerpo y puede llevar a una enfermedad tal como el cáncer.

- **Secuenciación de ADN** Mediante la secuenciación de ADN se busca determinar el orden exacto de las bases en una cadena de ADN. Como las bases existen como pares, y la identidad de una base está determinada por la otra base, los investigadores no necesitan reportar las dos bases que conforman el par.
  La secuenciación de ADN puede utilizarse para encontrar mutaciones o variaciones genéticas, que pueden jugar un rol importante en el desarrollo de enfermedades. La causa de la enfermedad puede estar ligada a un cambio tan pequeño como la sustitución, eliminación o adición de un par de bases, o tan grande como la eliminación de miles de bases.

- **Tandem Repeat** Un tandem repeat es una secuencia de dos o más pares de bases que se repite de tal manera que las repeticiones son adyacentes entre sí en el cromosoma. Estas secuencias también son llamadas VTNRs (variable number tandem repeats), debido a que distintos individuos de una población pueden tener cantidades de copias diferentes para cada tandem repeat.

3. Estado del arte

Existen distintos algoritmos desarrollados hasta el momento que se encargan de identificar TR en una secuencia. Todos ellos sufren de distintas limitaciones en cuanto a complejidad temporal, complejidad espacial, o alcance. Por ejemplo, existen algoritmos enfocados en la detección de secuencias de tamaño pequeño, otros que necesitan que no existan más de K variaciones en las repeticiones, otros que necesitan que se les especifique el tamaño de la secuencia repetida a encontrar, entre otras limitaciones.
Un grupo de algoritmos utilizados en este problema, es el basado en el cálculo de matrices de alineación. En este tipo de algoritmos se busca alinear subcadenas en una cadena y calcular un puntaje de alineación. De esta manera se encuentra la región de repetición más larga en una cadena. Este tipo de algoritmos tienen complejidad temporal de $O(n^2 log^2 n)$ y espacial de n^2, siendo n el largo de la secuencia. Sin embargo, puede verse que para cadenas muy largas, el algoritmo no sería útil pues su complejidad temporal es significativa.

Otro tipo de algoritmos utiliza el campo de la compresión de datos. En este tipo de algoritmos se propone que los patrones repetidos pueden ser descubiertos al codificar la información a números pequeños de bits. En estos casos se aplica un algoritmo de programación dinámica para compresión de datos, que permite calcular el largo mínimo de codificación de secuencias en tiempo linear. Sin embargo, este tipo de algoritmos se enfocan en encontrar secuencias, más no buscan encontrar patrones repetidos.

Con el fin de mejorar el desempeño en cuanto a complejidad temporal y espacial de los algoritmos ya desarrollados, se propuso el algoritmo basado en tuplas TRF [5], que es uno de los más reconocidos y utilizados en la actualidad. Este algoritmo busca mapear los patrones a tuplas y compararlas entre sí, de manera que no se tengan que utilizar matrices que requieran tiempos de procesamiento cuadráticos. Este algoritmo se encuentra disponible en línea para su utilización, con el nombre de *Tandem Repeats Finder*, y puede ser encontrado en *https://tandem.bu.edu/trf/trf.html*.

En el algoritmo de TRF, se modela la alineación de dos copias de tamaño n con ensayos de Bernoulli. De esta manera, se alinean dos secuencias y se indica con H si las dos bases son iguales, y con T se indica que no son iguales. Esto se puede observar en la figura 3.

Figura 3: Secuencias de ADN superpuestas

Es a partir de alineaciones que se consigue identificar los TR en secuencias. El algoritmo busca secuencias de largo k que estén separadas por una distancia común d. Para que estas dos secuencias hagan match, se busca que al alinearlas generen una secuencia HT con k cabezas (Head, de allí proviene la H). De esta manera, se busca una cadena de Hs de largo k para decir que dos secuencias de largo k hacen match.

El funcionamiento a grandes rasgos de la detección de secuencias repetidas se hace así: si se tiene una secuencia S, esta se escanea utilizando ventanas de tamaño k. La ventana se va corriendo a medida que se escanea. En la figura 4 se puede observar una ilustración de lo descrito.

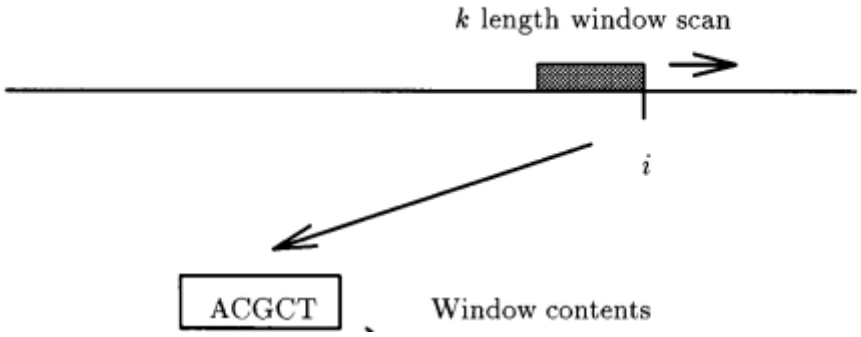

Figura 4: Escaneo de la secuencia con una ventana de tamaño k

Inicialmente se tiene toda la lista de secuencias de largo k que se pueden generar, que para el caso del alfabeto de ADN es 4^k, debido a que se tienen las 4 letras A, C, G, T. A esta lista de secuencias se le llama pruebas. De esta manera, si ocurre una prueba en el scan actual, se guarda la posición i en donde ocurrió la prueba. Para cada prueba p se mantiene una lista H_p de las posiciones en las que p ocurre en la secuencia S. Este funcionamiento puede ser observado en la figura 5.

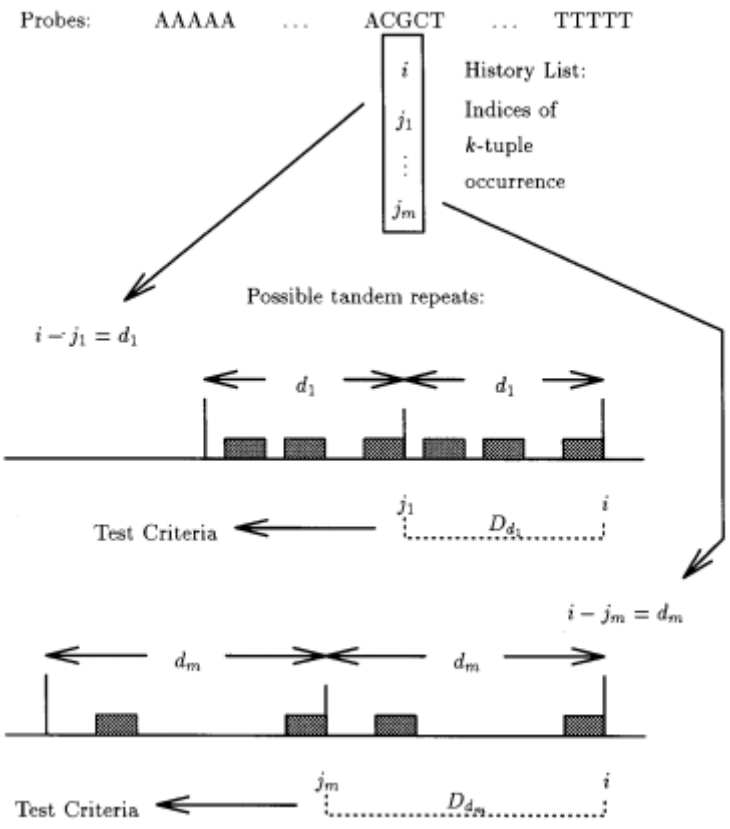

Figura 5: Detección de repeticiones de una ventana

Para la lista de ocurrencias de esa ventana se tienen las posiciones en las que se encuentra la secuencia. Si se tiene la posición i y j se tiene una distancia d al restar $i - j$. De esta manera y de acuerdo a la definición anterior para la alineación de dos secuencias, se buscan otras ocurrencias de la prueba que se encuentren a la misma distancia d en todas las ocurrencias en la lista de historial.

Si la información en la lista de distancias generada a partir del algoritmo descrito pasa una serie de criterios (que pueden ser configurados por el usuario) entonces se realiza el alineamiento de la secuencia con sus alrededores utilizando programación dinámica. Si por lo menos dos copias del patrón están alineadas con la secuencia, entonces se considera que se encontró un TR y se reportan las características de este. Cabe aclarar que en la detección y alineación se utilizan varios conceptos estadísticos que no fueron explicados debido a la complejidad de estos. Sin embargo, si se requiere información más detallada se puede consultar la referencia [9].

Similarmente, existe el algoritmo $dot - plot$ [6] que busca comparar una secuencia con si misma, con herramientas visuales y una representación distinta de las matrices que se venían utilizando, que permite la identificación de los TR en una secuencia.

4. Motivación

En la actualidad ya existen herramientas computacionales que permiten la identificación de los TR en secuencias de ADN. Sin embargo, algunas de estas herramientas fueron desarrolladas hace una cantidad considerable de tiempo, por lo cual son difíciles de adaptar a las tecnologías más modernas, como por ejemplo es el caso de Tandem Repeats Finder [5], desarrollada en 1999. Aunque existen herramientas más modernas, como por ejemplo *Dot2dot* [6], se observó que estas herramientas están desarrolladas para correr mediante línea de comandos, es decir, no ofrecen una interfaz de usuario que permita su uso de manera visual. Además, al ser herramientas desarrolladas bajo un contexto específico, son poco extensibles, y no permiten una interoperabilidad con otros sistemas que puedan requerir usarlas.

Teniendo en cuenta las complicaciones encontradas con las herramientas actuales para la detección de TR en secuencias de ADN, se identifica entonces la necesidad del diseño e implementación de una herramienta para identificar estas secuencias que sea acorde a la tecnología actual, que ofrezca una interfaz de usuario, y que sea flexible en su uso para permitir su uso por otros sistemas. En este caso, se busca específicamente que la herramienta pueda ser integrada con el sistema NGSEP [12], desarrollado por el grupo de investigación del Profesor Jorge Duitama de la Universidad de los Andes.

5. Resultados

La herramienta implementada consta de varias fases, cuya implementación tiene distintos detalles. A grandes rasgos, la herramienta hace un proceso de 3 fases: selección de candidatos, limpieza de candidatos redundantes, y alineación de candidatos. El diagrama de flujo que se muestra en la figura 6 detalla el funcionamiento general del algoritmo.

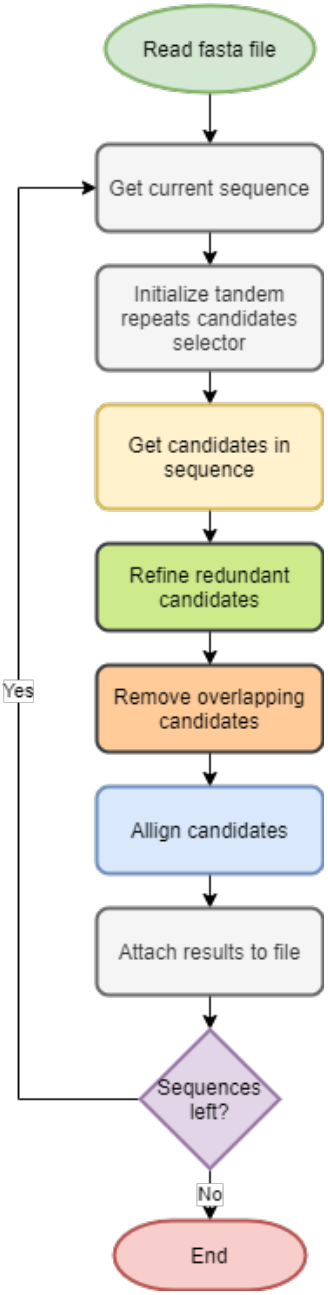

Figura 6: Diagrama de flujo general

El diagrama de clases de la herramienta implementada se detalla en la figura 7.

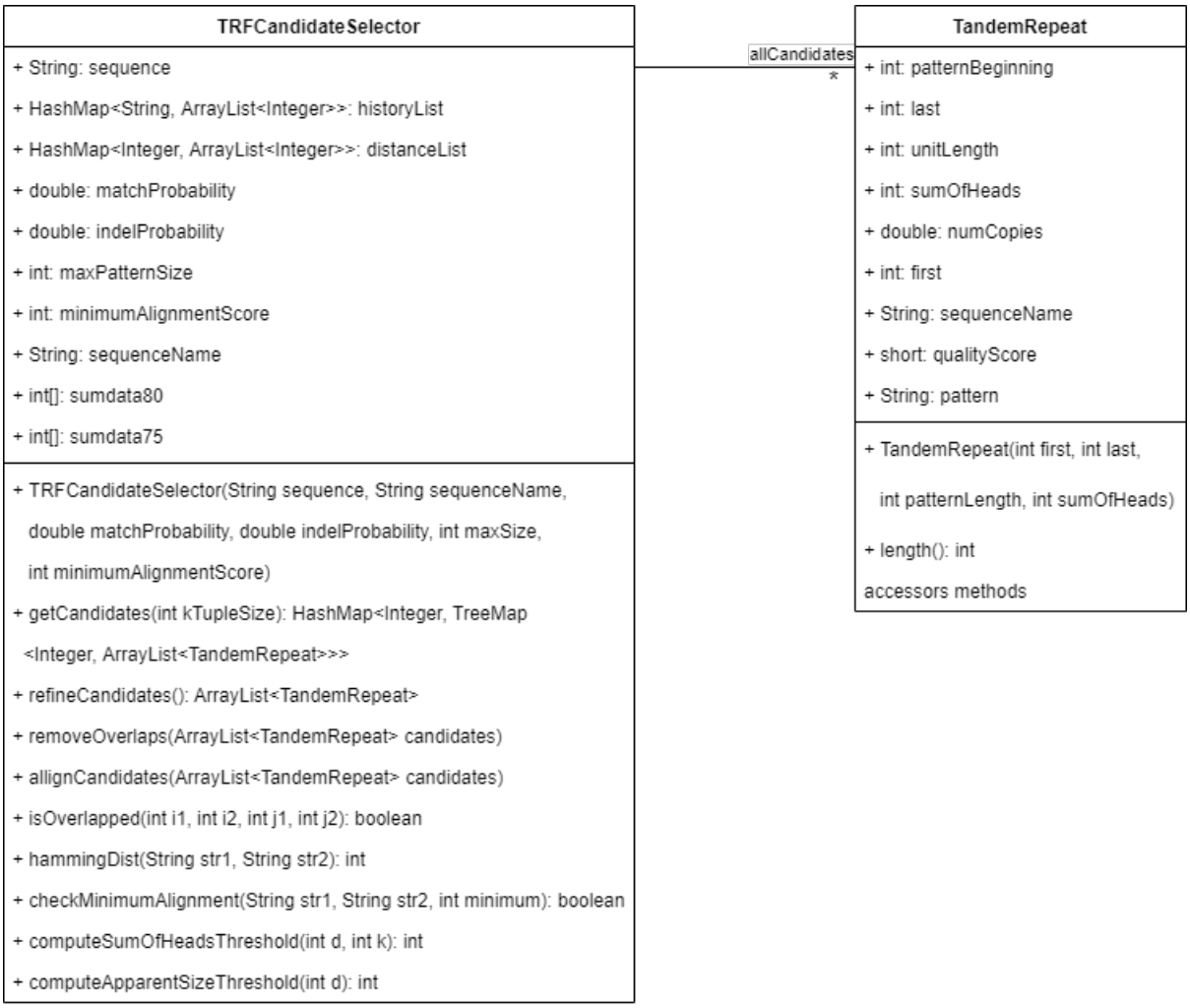

Figura 7: Diagrama de clases UML

Para implementar la herramienta inicialmente se trabajó con la documentación de TRF para la fase de selección de candidatos. Para esto se siguió el algoritmo inicial de esta herramienta. Se recorre la secuencia original seleccionando sub-secuencias de tamaño k. Para cada secuencia encontrada se guarda una lista de ocurrencias, en la que se guardan los índices en los que termina esa secuencia a lo largo de la secuencia original. Se calcula la distancia entre las dos secuencias como $i - j0$. Si esta distancia es menor que la definida como máxima, entonces se agrega el índice i a una lista que se tiene para la distancia calculada d0. Para cada distancia se tiene una lista que se comporta como una ventana. Los índices que sucedan antes de j0 no se consideran. Para cada lista de distancia se calcula el criterio de *sum of heads*, definido al inicio de este documento. Esto se calcula a partir de la cantidad de índices presentes en la lista de la distancia actual como *número de índices*tamaño k*. Esto se utiliza para tener una cantidad suficiente de bases correspondientes, de manera que el tandem repeat se pueda considerar como candidato.
Adicionalmente se calcula el criterio de tamaño aparente o *apparent size*. Esto se realiza debido a que el TR encontrado debe abarcar una cantidad de la distancia calculada para ser considerado como tal. En la figura 8 se ilustra esto. La primera secuencia podría considerarse como un TR. Sin embargo la segunda no podría ser considerada como un TR, pues esta no cubre una cantidad considerable de la distancia.

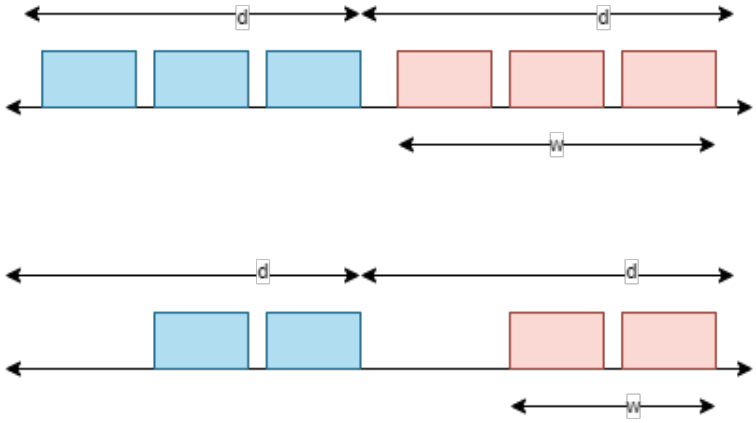

Figura 8: Criterio de tamaño aparente

A los candidatos se los pasa por ambos criterios, y si pasa en ambos el TR es añadido a la lista.

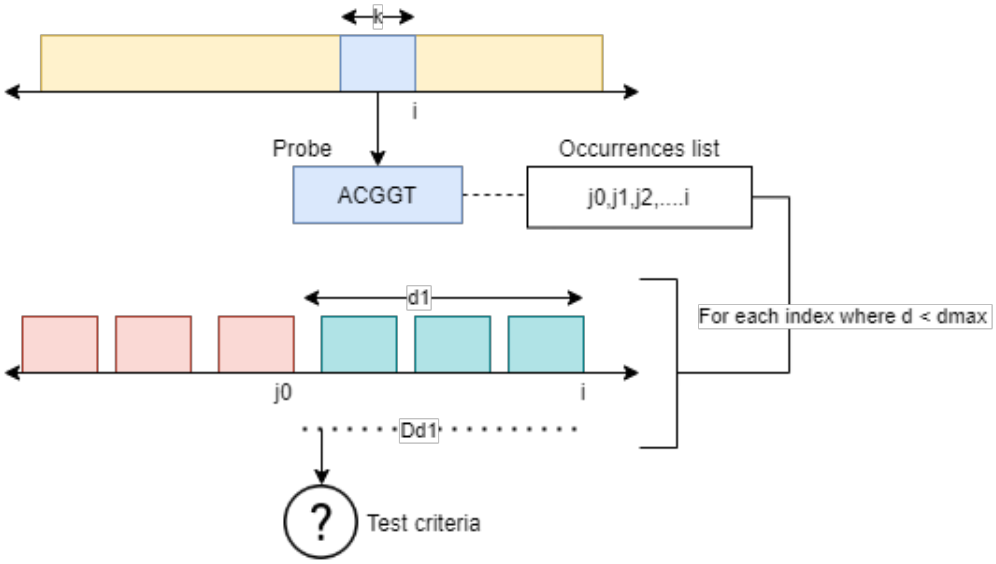

Figura 9: Algoritmo de selección de candidatos

El siguiente diagrama de flujo ilustra el funcionamiento de la selección de candidatos.

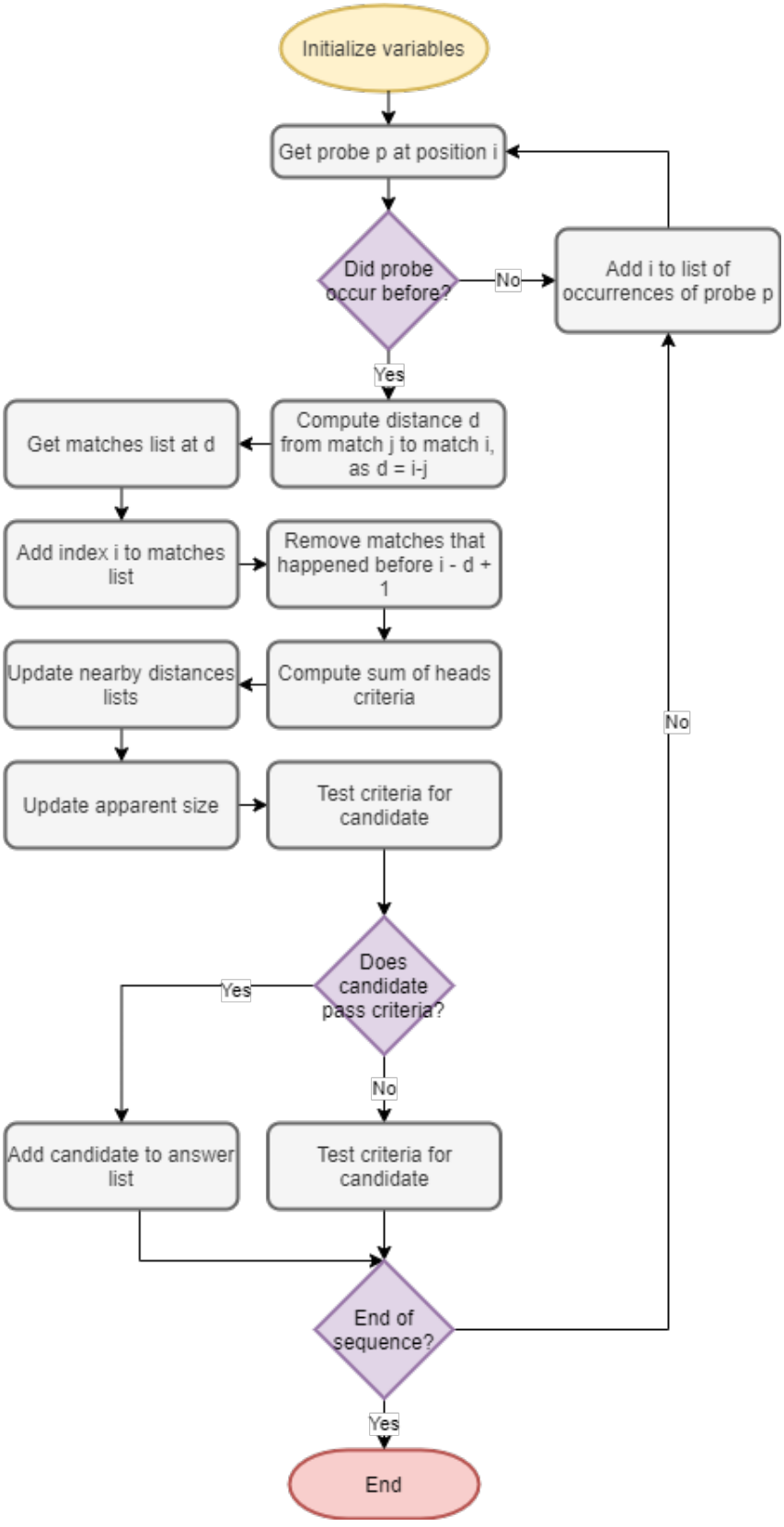

Figura 10: Diagrama de flujo selección de candidatos

Una vez se han encontrado los candidatos, se encuentran candidatos redundantes como el de la figura 11. En este caso el algoritmo detecta el mismo TR pero para distintos patrones que están superpuestos y tienen el mismo tamaño. Debido a esto y para no llenar el archivo de salido con TRs redundantes, se diseño un método que a partir de estos patrones calcula el número de veces aproximado que el patrón se repite teniendo en cuenta los TRs sobrelapados, y elimina los que estén

por detrás del que tenga el mayor índice final. Esto hace parte de la fase de limpieza de candidatos.

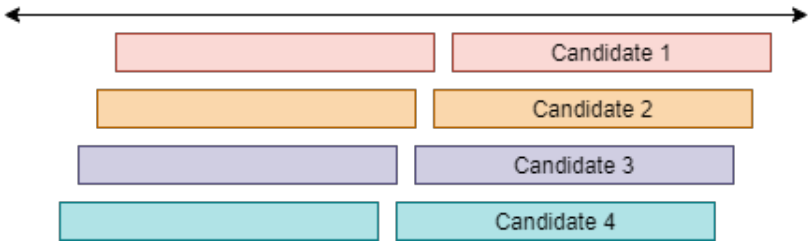

Figura 11: Situación en la que se deben refinar los candidatos

Para detectar esta situación se guardaron los TRs por distancia. De esta manera es posible recorrer todos los TRs a una distancia d sin tener que buscar en una lista todas las veces. Los TRs están ordenados por índice final. De esta manera si se obtiene el de mayor índice, se obtiene el TR que está ubicado más a la derecha. Para ese TR se calcula un rango en el que, de haber otro candidato en ese rango, se considera que ese TR tiene otra repetición. Esto puede observarse en la figura 12. En este sentido el TR de mayor índice sería el candidato 1. Los candidatos 2 y 3 al estar fuera del rango serían eliminados de la lista, pues no proveen información adicional. Si hay algún candidato dentro del rango aceptable y con la misma distancia se considera que se encontró otra copia del patrón. Se actualiza la cantidad de copias y se elimina el candidato. De esta manera cuando ya no hayan más candidatos en rango ya se tiene solo la copia roja con su número de repeticiones actualizado.

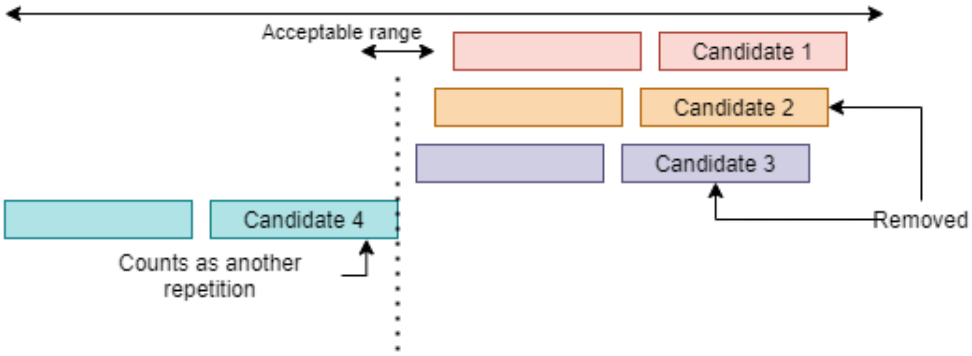

Figura 12: Rango para refinar candidatos

En la figura 13 se puede observar el diagrama de flujo que explica el funcionamiento del método.

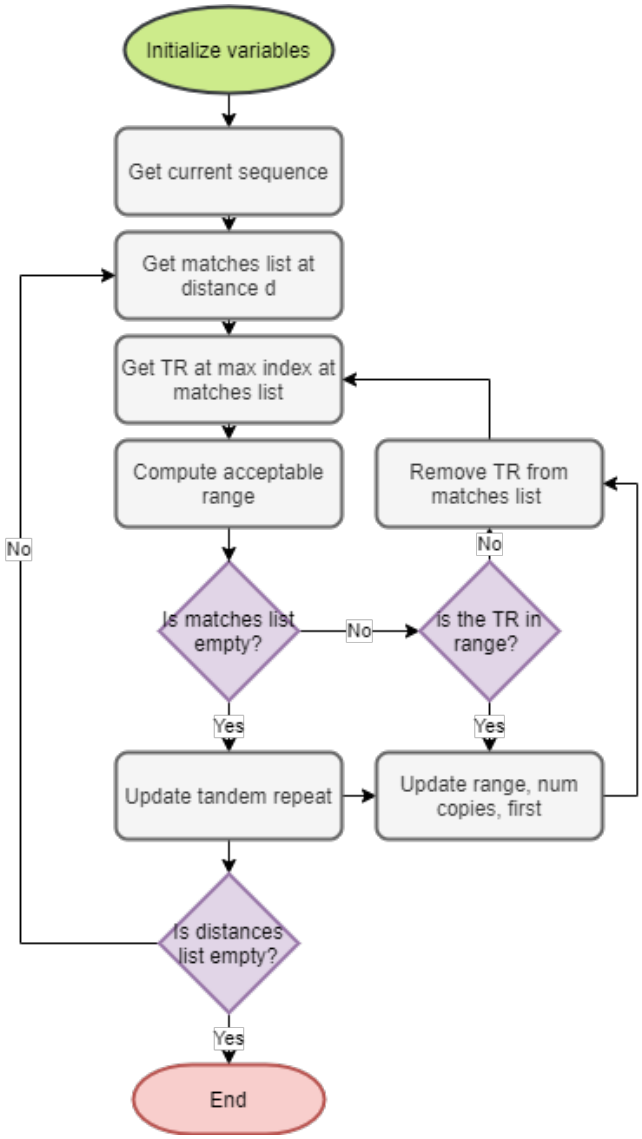

Figura 13: Diagrama de flujo refinación de candidatos

Una vez los candidatos son refinados, existe otra situación a considerar. Véase la figura 14, en la cual se encuentran TR para la misma región pero con distinto número de copias y periodo.

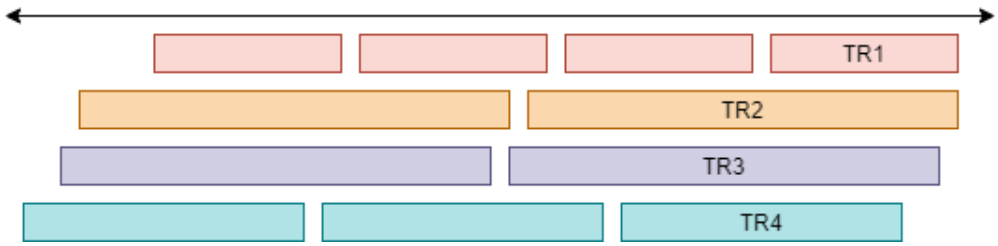

Figura 14: Sobrelape entre candidatos

En este caso también se definió un método que elimina estos sobrelapes y conserva el candidato que cubra un mayor tamaño en la secuencia. Este método primero ordena los candidatos por el primer índice. Se tiene una referencia a la región, que va desde el menor índice encontrado hasta el mayor índice encontrado. Inicialmente esa referencia son los índices del primer candidato encontrado. Para los siguientes se verifica si existe un sobrelape entre índices. Si es así, se actualiza la referencia de ser necesario, y se verifica si el candidato actual cubre una mayor distancia. Si es así se actualizan las referencias para tener el mejor. Si no es así, este candidato se elimina de la lista. Si el candidato requerido no hace sobrelape, significa que ya se pasó a otra región distinta, por lo que las referencias

se reinician. En la figura 15 se puede observar el flujo del método.

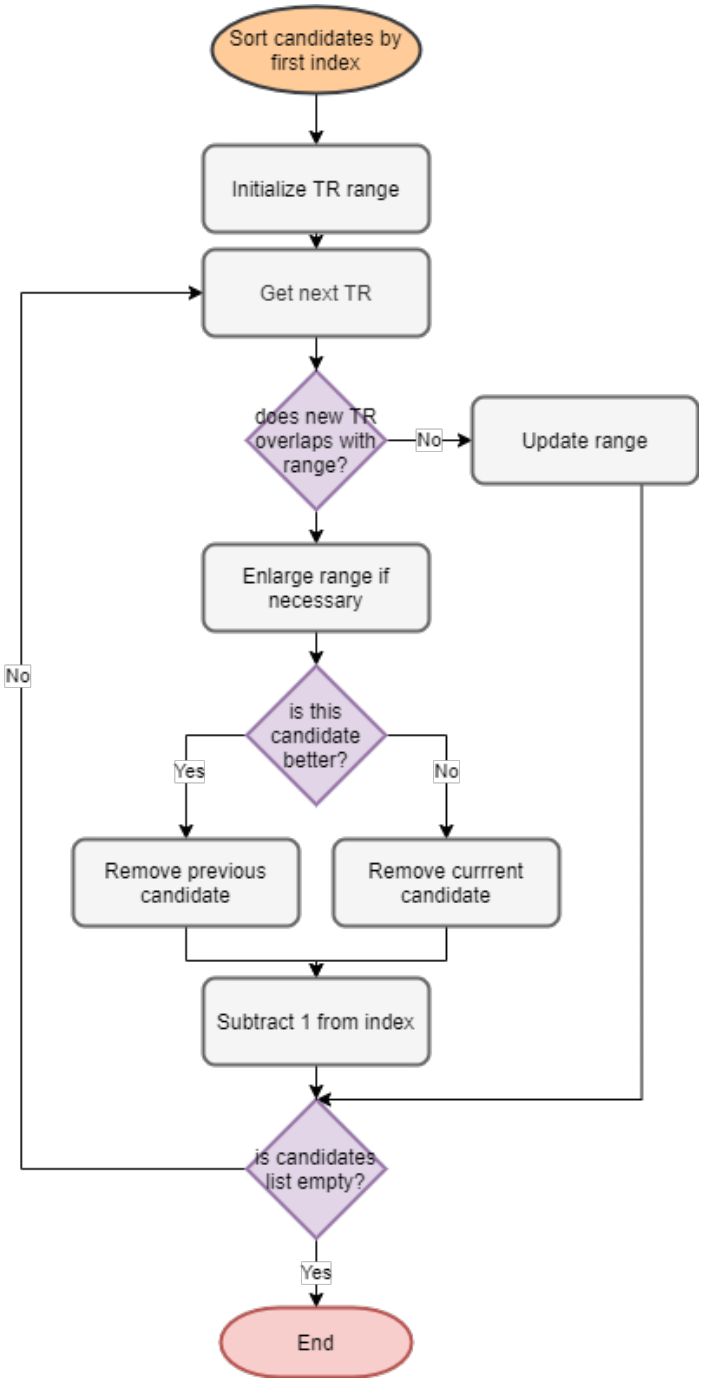

Figura 15: Diagrama de flujo eliminación de sobrelapes

Una vez los candidatos han sido filtrados, se procede a hacer los procesos de alineamiento para realizar el último filtro. Al ser el alineamiento pareado una tarea costosa en recursos, esta se realiza sobre los candidatos filtrados para ahorrar tiempo. En este caso se obtiene un candidato. Este candidato es dividido en partes si se identifica que se trata de un patrón repetido en el patrón identificado. Posteriormente se hace un proceso de corrección, en el que se buscan copias que no hayan sido detectadas en los pasos anteriores. Para esto, se alinea el patrón encontrado con la secuencia anterior al candidato. Si alinear estas dos secuencias provee secuencias alineadas con pocas diferencias, se actualiza el número de copias y el índice de inicio. Esto también se realiza hacia adelante, alineando el patrón con la secuencia posterior al último índice. Con esto es posible encontrar copias faltantes.
Una vez se tiene el número de copias y el patrón, este patrón se multiplica N veces y se alinea con el candidato encontrado. Si la alineación pasa un puntaje definido, se reporta el candidato. El

puntaje se calcula como +2 si hay un match en la alineación, y −4 si hay un − o si los caracteres son distintos. Sin embargo estos valores son ajustables. En la figura 16 se observa el flujo del método diseñado.

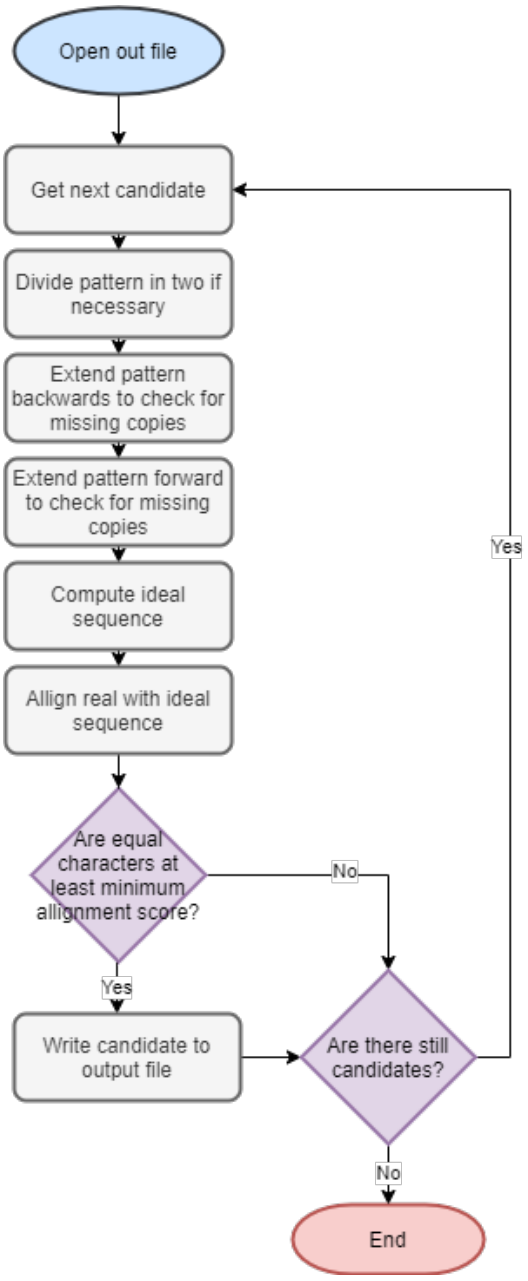

Figura 16: Diagrama de flujo alineación de candidatos

La herramienta es capaz de procesar archivos multiFASTA. La salida de la herramienta es un archivo TXT en el que en cada línea se representa, en orden:

- Nombre de la secuencia
- Índice de inicio
- Índice de fin
- Periodo
- Número de copias
- Tamaño total
- Patrón

- Secuencia total
- Secuencia ideal (patrón repetido N veces)
- Resultado de la alineación de ambas secuencias
- Puntaje de alineamiento del TR encontrado

Para probar la herramienta implementada se desarrolló un simulador de secuencias, que simula una secuencia que contiene TRs. En la sección de métodos se detalla la implementación de este simulador. Inicialmente se probó con una secuencia de un largo aproximado de 300kbp. Esta secuencia contiene 199 tandem repeats. Se calculó precisión y recall para cada herramienta (los detalles se pueden consultar en la sección de métodos) y se obtuvieron los siguientes resultados:

Cuadro 1: Comparación de herramientas

Source	Number of TRs	Precision	Recall	F1 Score
TRF	158	98.7629 %	94.4731 %	96.5703 %
mREPS	562	26.9205 %	83.8995 %	40.7618 %
Sciroko	45	99.9997 %	20.6695 %	34.2580 %
ATRHunter	372	46.4193 %	86.3782 %	60.3869 %
MyTRDetector	174	96.1432 %	84.7233 %	90.0727 %

Los resultados fueron graficados para observar el contraste entre herramientas:

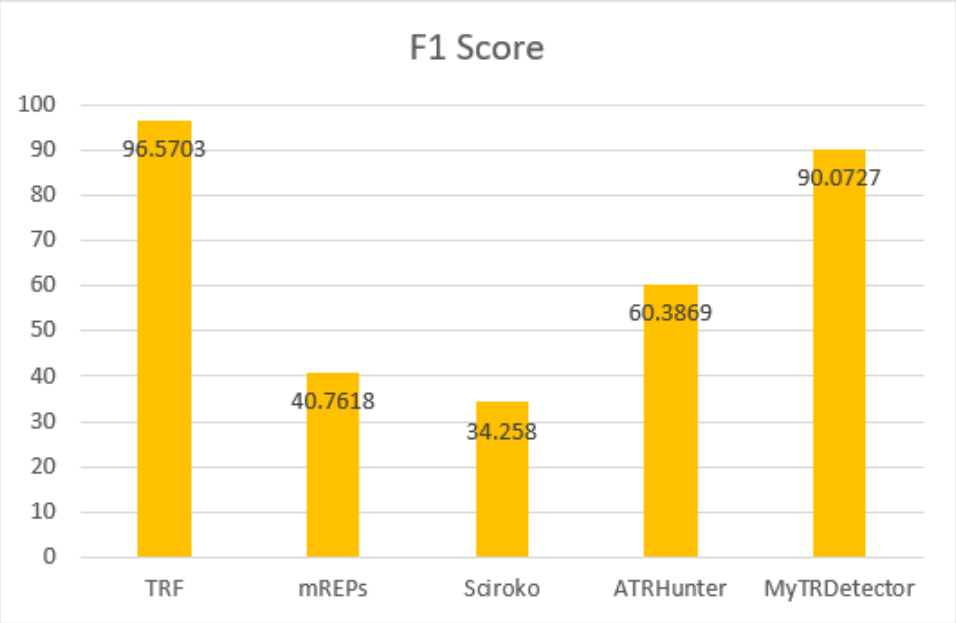

Figura 17: Puntaje F1 para cada herramienta

De todas las herramientas evaluadas, se identificó que TRF fue el mejor en términos de precisión y recall, lo que se refleja en el puntaje F1. Se puede identificar que la herramienta es muy buena encontrando los TRs originales, y que no genera muchos TRs adicionales. De todas las bases presentes, esta herramienta encontró aproximadamente el 96 %, lo cual es excelente.
Por otra parte la herramienta mREPS encuentra aproximadamente el mismo porcentaje de TRs, pero encuentra muchos más TRs no reportados. Esto no necesariamente es malo, puede deberse a métodos de filtrado diferentes o a reportes en donde haya redundancia. Sin embargo, esto baja su puntaje F1, haciéndola la segunda peor herramienta de las probadas para el set de prueba.
Sciroko solo permite patrones de largo máximo 6, por lo cual era esperable que tuviera una alta precisión: es capaz de identificar correctamente todos los TRs que identifica. Sin embargo, esto implica un recall bajo, pues solo es capaz de encontrar un 20 % de todos los TRs existentes. Esto provoca que su puntaje F1 sea bastante bajo, haciéndola la última herramienta entre todas las probadas.
Por último la herramienta ATRHunter puede identificar aproximadamente un 86 % de todos los

TRs, pero genera algunos adicionales lo que hace que su nivel de precisión baje. Sin embargo, la herramienta posee buenos resultados y un puntaje F1 de 60 %.

Debido a estos resultados, la implementación se basó en la herramienta TRF, al ser esta la más completa, la más documentada, y la que pose el código utilizado en Github https://github.com/Benson-Genomics-Lab/TRF.

En cuanto a la herramienta implementada, MyTRDetector desde ahora, se obtuvieron buenos resultados en las métricas probadas. En cuanto a la precisión, aunque la herramienta no logra superar a TRF, tiene un resultado muy cercano, pues de todos los TR identificados, 96 % son reales. En términos de recall, la herramienta es capaz de identificar 84 % de los TRs en el archivo original, dando un valor similar al de TRF. Cabe aclarar que los resultados siempre estarán sujetos a la secuencia en la que se ejecute y a los parámetros de cada herramienta. En términos del puntaje F1 fue la segunda mejor herramienta después de TRF.
Para la prueba mostrada se utilizaron los siguientes parámetros:

- Probabilidad de match de 0.8
- Probabilidad de indel de 0.1
- Máximo periodo a identificar de 25
- Mínimo puntaje de alineamiento de 50
- Puntaje de match de 2
- Penalización por miss de 4

Con el fin de observar más a detalle el comportamiento de TRF contra MyTRDetector, se utilizó el puntaje de alineamiento como quality score. Este es un parámetro que en ambas herramientas sirve para filtrar elementos. Para ambas se detalla un puntaje mínimo, de manera que solo los TRs que tengan un puntaje de alineamiento superior al indicado son reportados. Aunque ambas herramientas calculan el puntaje de manera distinta, se graficó la variación de los parámetros de precisión y recall para distintos parámetros de alineamiento:

Cuadro 2: Rendimiento de herramientas bajo distintos puntajes de alineamiento

	TRF		**MyTRDetector**	
Minimum Alignment Score	**Precision**	**Recall**	**Precision**	**Recall**
10	68.7044	97.0711	82.0256	92.7085
20	80.1063	97.4976	86.6139	92.7085
30	95.4176	96.6089	93.0158	92.7085
35	97.1607	95.8924	94.7443	92.4708
40	98.361	95.2526	94.9722	92.2969
50	98.7629	94.4731	96.0892	91.1488
80	99.0415	90.4298	96.4864	86.1832
100	99.0415	90.4298	96.8631	81.0206
120	99.2501	80.4975	96.9182	78.154
200	99.3417	56.9028	98.2501	57.3301
300	99.362	32.4925	98.6169	36.2732

Se pueden graficar los resultados para observar el rendimiento de ambas herramientas:

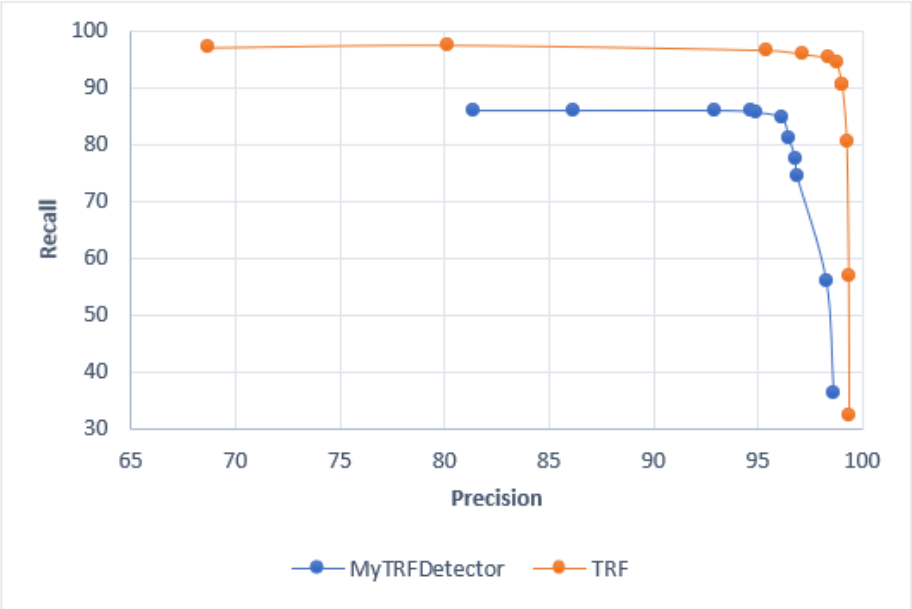

Figura 18: Variación de precision y recall bajo distintos puntajes de alineamiento

En la figura se puede observar que efectivamente TRF tiene un mejor rendimiento para la variación de parámetros. Sin embargo, la herramienta desarrollada también logra detectar correctamente los TRs en el archivo de prueba, y obtiene buenos valores que están muy cerca a los de TRF. Sin embargo para ambas herramientas se puede observar el comportamiento esperado: si el puntaje mínimo es muy bajo, los valores de recall son altos, pues se identifica una mayor número de TRs, pero esto baja la precisión al indentificarse más falsos positivos. Si el puntaje mínimo es muy alto, se restringe mucho la salida, lo que provoca que el recall baje mientras que la precisión sube. Se busca utilizar la herramienta en los puntos de la curva en la que ambos valores sean altos.

Para probar el rendimiento de la herramienta en un set mas grande este se corrió con el genoma de levadura. El tiempo de ejecución de la herramienta fue de $6220ms$, aproximadamente 6s. La salida de la herramienta se comparó con la salida de TRF como goal standard.

Cuadro 3: Precision and recall para el genoma de levadura

Minimum Alignment Score	**Precision**	**Recall**	**F1**
10	22.8303	76.475	35.16322
20	28.3382	75.2596	41.17311
25	35.599	73.9407	48.05956
30	42.7929	72.5468	53.83208
35	53.3045	69.5343	60.34724
37	56.5093	67.9943	61.72208
40	59.9616	66.2163	62.93393
50	67.2686	54.6385	60.29928
80	71.6604	26.3117	38.49069
90	71.6178	21.0211	32.50222

También se calculó el puntaje F1 para poder obtener el valor del parámetro que calcula los mejores resultados. Los resultados fueron graficados para observar el comportamiento de la herramienta:

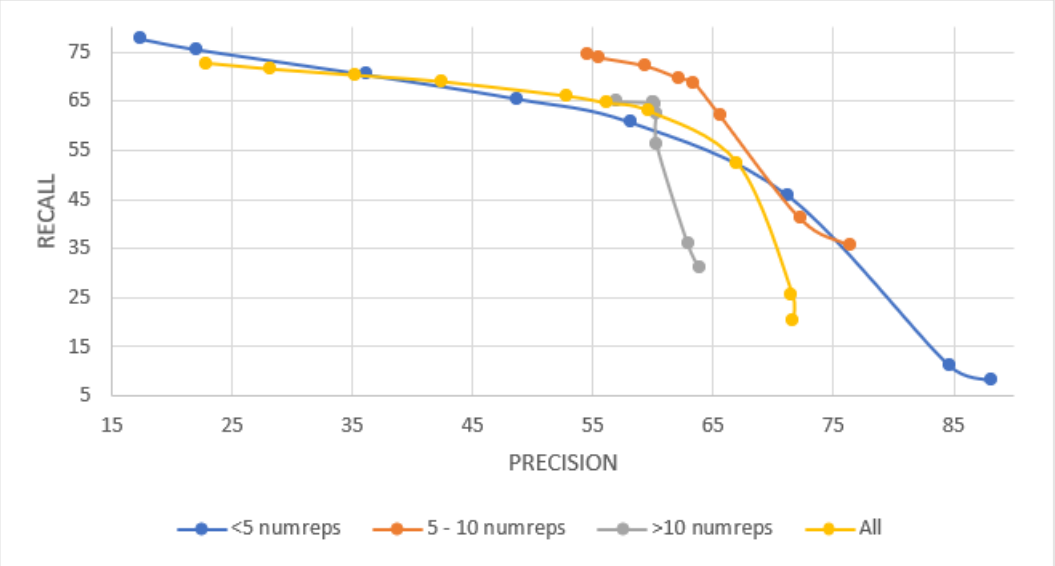

Figura 19: Variación de precision y recall bajo distintos puntajes de alineamiento

De acuerdo a la gráfica y al puntaje F1, podemos observar que el mejor rendimiento se da con un puntaje de alineamiento mínimo de 40. En este caso se puede observar un menor rendimiento que en el del primer caso de prueba, pues se está comparando contra TRF y es esperable que ambas herramientas produzcan resultados diferentes. En el mejor punto se obtiene una precisión de alrededor de 60 % y un recall de 66 %. La herramienta es capaz de identificar correctamente una gran cantidad de TRs, pero produce un poco más de los que produce TRF para este archivo, lo que hace que su precisión no sea muy alta.

Con el fin de perfilar más la herramienta se discriminaron los resultados por número de copias. De acuerdo a los resultados obtenidos se puede observar que esta tiene un buen comportamiento con TRs que se repiten de 2 a 10 veces. Sin embargo, se observó que la mayor debilidad de esta se encuentra en los TRs que poseen un número de repeticiones mayor a 10. Esto se debe principalmente a que la herramienta no tiene herramientas de filtrado tan sofisticadas, lo que puede que no le permita llegar al mejor patrón de entre todas las posibilidades. Esto en otros términos lleva a que TRs de gran tamaño puedan ser reportados con menor número de copias, lo que produce que se reporten desfasados y con menor número de copias.

6. Métodos

6.1. Script simulador

Para probar las herramientas existentes y la desarrollada es necesario contar con un set de prueba, del que se conozca dónde están ubicados los Tandem Repeats, las características de cada uno, y cuántos hay en el archivo original. Para esto se desarolló un script en Python 3 en el que se define el largo del archivo en cantidad de bases, y cuántos tandem repeats se quieren insertar. El script genera una secuencia inicial de largo L utilizando el alfabeto ACGT. Posteriormente se van generando los tandem repeats uno por uno hasta que se tenga la cantidad deseada N. Para cada tandem repeat se define un tipo que se define aleatoriamente:

1. TR puro: se trata de un tandem repeat sin alteraciones. Se genera un patrón de largo aleatorio, y este se repite en un número de repeticiones entre 2 y 20.

2. TR con bases cambiadas: se trata de un tandem repeat en el que de 1 a $periodo/2$ bases son reemplazadas por una distinta. El periodo y el número de repeticiones es definido inicialmente y posteriormente se cambian las bases.

3. TR cortado: en esta variación se genera un TR de la misma manera que los anteriores, pero se cortan de 1 a $periodo/2$ bases al final.

4. TR con indels: en esta variación se genera un TR de la misma manera que los anteriores, pero se insertan o se borran 1 a *periodo*/2 bases en posiciones aleatorias.

Posterior a la creación de cada tandem repeat, estos son insertados uniformemente en la secuencia generada al inicio. En cada iteración se van actualizando los índices de referencia para tener los índices de inicio y fin de cada TR. En la figura 20 se observa como se realiza la inserción de los TR:

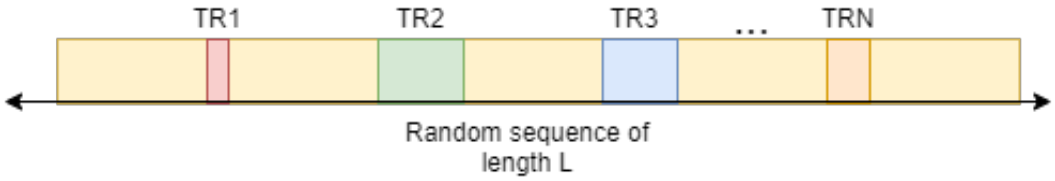

Figura 20: Inserción de Tandem Repeats en script de simulación

Cada repetición es guardada en un archivo JSON para mantener información adicional de cada repetición. Para cada TR se tiene una estructura similar a la siguiente:

```
    {
        "Periodo": 14,
        "Patron:": "TCTCAAGTTTGGTC",
        "# Rep": 2,
        "Tamanio total": 28,
        "seq": "TCTCATGTTTGGTCTTCCAAGTTTGGTC",
        "indice inicial": 43094,
        "indice final": 43121,
        "Tipo alteracion": "alteraciones",
        "Informacion adicional": [
            {
               "Cadena original":"TCTCAAGTTTGGTCTCTCAAGTTTGGTC",
               "Cadena nueva":"TCTCATGTTTGGTCTTCCAAGTTTGGTC",
               "Numero de modificaciones": 3,
               "Modificaciones": [
                    {
                        "Indice": 43109,
                        "Original": "T",
                        "Nuevo": "C"
                    },
                    {
                        "Indice": 43098,
                        "Original": "A",
                        "Nuevo": "T"
                    },
                    {
                        "Indice": 43108,
                        "Original": "C",
                        "Nuevo": "T"
                    }
                ]
     }
```

Por último y para poder comparar con otras herramientas, se generó un formato línea por línea en el cual se muestra por línea el índice de inicio, fin, periodo, número de repeticiones, tipo de alteración, largo de la secuencia, patrón y secuencia total. El archivo luce así:

```
38119 38158 4 10 Puro 40 ATTC ATTCATTCATTCATTCATTCATTCATTCATTCATTCATTC
39659 39928 18 15 Puro 270 CCCTATTAGTGTCCGATC ...
41429 41593 15 11 alteraciones 165 ATTGTTAATTTGAAG ...
```

El archivo JSON se realizó para poder tener un mayor detalle de los TR creados, mientras que el archivo de texto se realizó para poder comparar con otras herramientas y para tener una salida más compacta y para que este pueda ser leído por lenguajes de programación.

6.2. Pruebas de herramientas

Se probaron herramientas ya existentes de detección de TR para familiarizarse con su funcionamiento, y así poder identificar los principales problemas de implementación y uso que estas tenían. Se probaron las siguientes herramientas:

- TRF [18]
- Mreps [19]
- SciRoko [21]
- ATRHunter [22]

Se compararon las herramientas existentes contra el archivo de referencia generado. Se generó un archivo con 199 tandem repeats y de largo 300KB. El archivo se corrió en las 4 herramientas con los siguientes comandos:

- TRF: *trf409.dos64.exe seqB.fa 2 7 7 80 10 50 25 -d -m*
- MREPS: *mreps.exe -minp 2 -maxp 20 -res 10 -xmloutput mreps.xml -fasta seqB.fa*
- SciRok: Al ser esta herramienta de uso con interfaz no se ejecutó comando. Se definió un máximo periodo de 6, pues esta herramienta no permitió configurar un mayor tamaño.
- ATRHunter: *2 5 7 7 max 20 1 - 70*

Cada herramienta provee una salida con distintos formato. Estos formatos fueron leídos y procesados en un proyecto en Java que sigue la siguiente lógica:

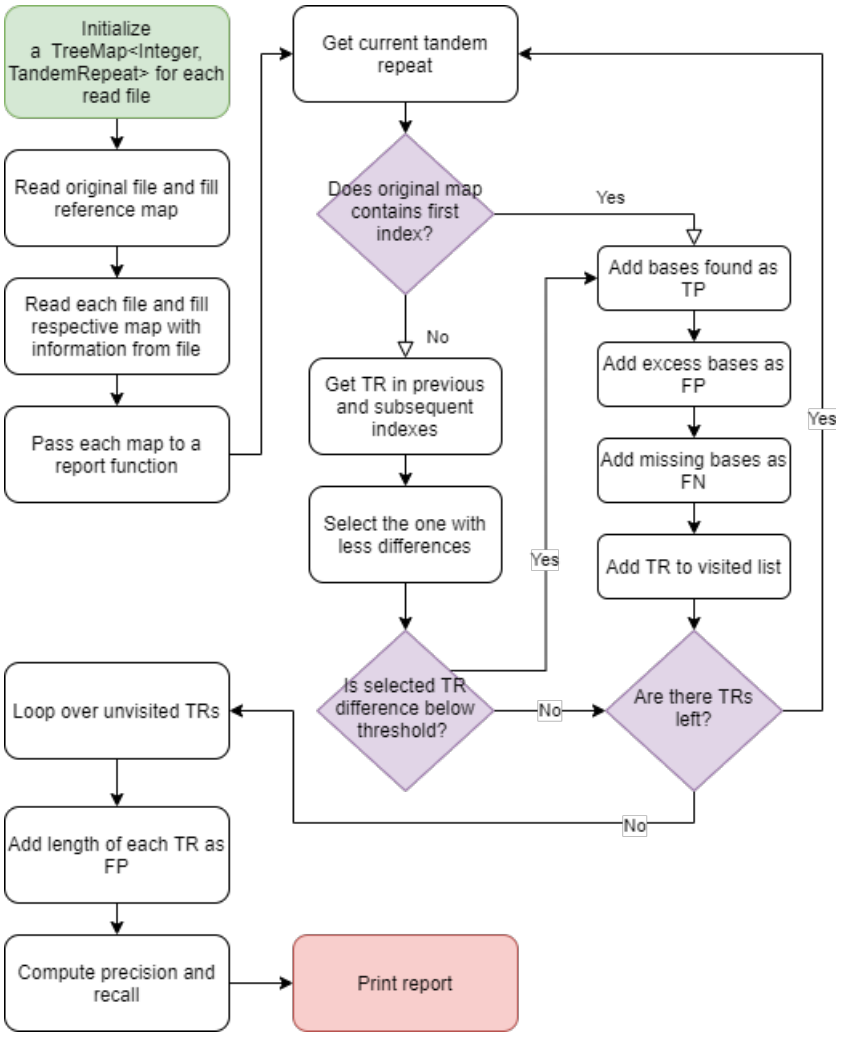

Figura 21: Diagrama de flujo comparación de herramientas

Para cada archivo producido por cada herramienta se crea un TreeMap, en el cual se guarda en la llave el índice de inicio, y en el contenido un objeto de tipo Tandem Repeat que almacena la información de cada TR. El TreeMap se utiliza para poder tener la estructura ordenada por índices de inicio. La lógica seguida es buscar en el mapa original el TR actual. Si se encuentra el TR se actualizan las bases encontradas. Si no, se busca en los índices anterior y siguiente. Si se cumple con un umbral definido, se considera que se encontró el TR. Si no se encuentra en ninguno de estos índices, se asume que es un TR identificado como falso positivo. Para manejar la cantidad de verdaderos positivos (FP), falsos negativos (FN) y falsos positivos (FP) se cuentan las bases encontradas en el rango original como TP. Las bases encontradas por fuera del rango, estén o no dentro de un TR son consideradas como FP, y las bases no encontradas en un TR encontrado son consideradas como FN. La figura 22 ilustra este comportamiento.

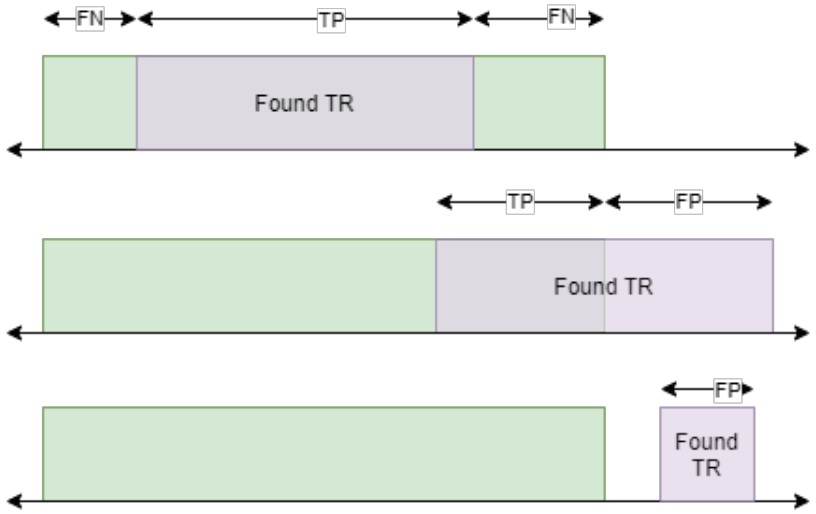

Figura 22: Superposición de bases

A partir de esto se calcula precisión y recall siguiendo las siguientes fórmulas:

$$Precision = \frac{tp}{tp + fp} \tag{1}$$

$$Recall = \frac{tp}{tp + fn} \tag{2}$$

En este caso se entiende la precisión como de los TR encontrados, cuántos realmente estaban en el archivo original. En el caso del recall se entiende como la capacidad de la herramienta para encontrar TRs.
También se calculó el puntaje $F1$, que combina las medidas de precisión y recall y nos ayuda a comparar distintas herramientas con una sola métrica. Este se calculó como:

$$F1 = 2 * \frac{precision * recall}{precision + recall} \tag{3}$$

7. Discusión

Fue posible implementar una herramienta capaz de detectar tandem repeats en secuencias de ADN. La implementación se basó en el software existente TRF, pero la selección y el filtrado de candidatos se realizó de manera diferente. Esto lleva a que ambas herramientas no reporten la misma cantidad de TRs, ni tengan el mismo rendimiento. Sin embargo se observó que ambas herramientas tienen buenos resultados en los datos utilizados para probar.
La herramienta TRF tiene un mejor rendimiento que la implementada en este proyecto. Sin embargo, debe considerarse que TRF es una herramienta ampliamente conocida que fué desarrollada por varias personas y que ha contado con un mayor tiempo de desarrollo. Sin embargo, la herramienta implementada, MyTRDetector, logra hacer una buena identificación de TRs, es desarrollada en Java, Open Source (disponible en https://github.com/mpfranco10/TandemRepeatsProject), es integrable con el software NGSep, y además es fácil de ejecutar y transparente al usuario.

El rendimiento de la herramienta depende de los parámetros brindados por el usuario. Entre más estrictos sean los parámetros, en especial el score de alineamiento, más TRs reporta la herramienta, aumentando la precisión pero disminuyendo el recall. Por el contrario mientras más lapsos sean los parámetros, el recall aumenta mientras que la precisión disminuye. El valor puede ser variado para elegir el resultado que más se adapte a lo que el usuario esté esperando de la herramienta.
Es posible en el futuro refinar aún más la herramienta, pues con la implementación realizada no se busca el mejor patrón sino que se utilizan métodos un poco menos complicados que los implementados en TRF. Esto provoca que TRF, tenga mejores resultados, pues posee métodos de filtrado y cálculos de métricas más avanzados. Para alcanzar o incluso superar el rendimiento de TRF es necesario realizar métodos de filtrados más exhaustivos y calcular más puntajes que den una visión de qué tan buena es una predicción.
Las pruebas se realizaron siguiendo un método de diagnóstico planeado, en el cual se beneficia o se castiga a una herramienta de acuerdo al número de bases que prediga correctamente. Esto permite realizar comparaciones más a fondo que si solo se compararan las herramientas de acuerdo a número de TRs encontrados, pues aunque una herramienta prediga más que otra no significa que esas predicciones sean correctas.

Referencias

[1] Jain, M., Koren, S., Miga, K. et al. Nanopore sequencing and assembly of a human genome with ultra-long reads. Nat Biotechnol 36, 338–345 (2018). https://doi.org/10.1038/nbt.4060

[2] National Human Genome Research Institute. (s.f.). A Brief Guide to Genomics. Obtenido de NIH: https://www.genome.gov/about-genomics/fact-sheets/A-Brief-Guide-to-Genomics

[3] Myers, P. (2007) Tandem repeats and morphological variation. Nature. Obtenido de Scitable: https://www.nature.com/scitable/topicpage/tandem-repeats-and-morphological-variation-40690/

[4] Jain, M., Koren, S., Miga, K. et al. Nanopore sequencing and assembly of a human genome with ultra-long reads. Nat Biotechnol 36, 338–345 (2018). https://doi.org/10.1038/nbt.4060

[5] Gary Benson, Tandem repeats finder: a program to analyze DNA sequences, Nucleic Acids Research, Volume 27, Issue 2, 1 January 1999, Pages 573–580, https://doi.org/10.1093/nar/27.2.573

[6] Loredana M Genovese, Marco M Mosca, Marco Pellegrini, Filippo Geraci, Dot2dot: accurate whole-genome tandem repeats discovery, Bioinformatics, Volume 35, Issue 6, 15 March 2019, Pages 914–922, https://doi.org/10.1093/bioinformatics/bty747

[7] Scott A. Jackson, A. I.-H. (27 de Junio de 2011). Sequencing crop genomes: approaches and applications. Obtenido de New Phytologist: https://nph.onlinelibrary.wiley.com/doi/full/10.1111/j.1469-8137.2011.03804.x

[8] Sergey Koren, B. P. (23 de Agosto de 2016). Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation. Obtenido de Genome Research: https://genome.cshlp.org/content/27/5/722

[9] Gary Benson, A space efficient algorithm for finding the best nonoverlapping alignment score, Theoretical Computer Science, Volume 145, Issues 1–2, 1995, Págs 357-369. Obtenido de https://doi.org/10.1016/0304-3975(95)92848-R.

[10] Milosavljević, A., & Jurka, J. (1993). Discovering simple DNA sequences by the algorithmic significance method. Computer applications in the biosciences : CABIOS, 9(4), 407–411. https://doi.org/10.1093/bioinformatics/9.4.407

[11] Gemayel, R. (Diciembre de 2010). Variable Tandem Repeats Accelerate Evolution of Coding and Regulatory Sequences. Obtenido de https://www.annualreviews.org/doi/10.1146/annurev-genet-072610-155046

[12] Tello Velasco, Daniel & Gil, Juanita & Loaiza Orduz, Cristian Dario & Riascos, John & Cardozo, Nicolás & Duitama, Jorge. (2019). NGSEP3: Accurate variant calling across species and sequencing protocols. Bioinformatics. 35. 10.1093/bioinformatics/btz275.

[13] Saeed, Abdullah & Wang, Rongzhi & Wang, Shihua. (2016). Microsatellites in Pursuit of Microbial Genome Evolution. Frontiers in Microbiology. 6. 1462. 10.3389/fmicb.2015.01462.

[14] LBI (2020). How does Tandem Repeats Finder work?. Obtenido de https://tandem.bu.edu/trf/trfdesc.html

[15] LANDAU, GAD M & SCHMIDT, JEANETTE P & SOKOL, DINA. (2001). An Algorithm for Approximate Tandem Repeats. Obtenido de http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.7.9814&rep=rep1&type=pdf.

[16] Sokol, Dina. (2009). DIMACS EDUCATIONAL MODULE SERIES Finding Repeats Within Strings. Obtenido de http://archive.dimacs.rutgers.edu/Publications/Modules/Module09-2/dimacs09-2.pdf.

[17] Angelika Merkel, Neil Gemmell, Detecting short tandem repeats from genome data: opening the software black box, Briefings in Bioinformatics, Volume 9, Issue 5, September 2008, Pages 355–366, https://doi.org/10.1093/bib/bbn028

[18] Tandem Repeats Finder. (s.f.). Obtenido de Boston University: http://tandem.bu.edu/trf/trf.html

[19] mreps. (s.f.). Obtenido de Bonsai Software: https://bioinfo.lifl.fr/mreps/mreps.php

[20] STAR. (s.f.). Obtenido de ATGC: http://atgc.lirmm.fr/star/

[21] SciRoko . (s.f.). Obtenido de Bioinformatics Robert Kofler: https://www.kofler.or.at/bioinformatics/SciRoKo/Download.html

[22] ATRHunter. (s.f.). Obtenido de Israel Institute of Technology: http://bioinfo.cs.technion.ac.il/atrhunter/ATRHunter.htm

[23] SSRIT - Simple Sequence Repeat Identification Tool. (s.f.). Obtenido de Gramene: https://archive.gramene.org/db/markers/ssrtool