ISSN 2007-9737

Evaluación de la influencia de los recursos computacionales en la variabilidad y calidad de ensamblaje *de novo* de transcriptoma

Patricia Carvajal López[1], Fernando D. Von Borstel Luna[1], Joaquín Gutiérrez Jagüey[1], Humberto Mejía Ruiz[1], Gabriela Rustici[2], Eduardo Romero Vivas[1]

[1] Instituto Politécnico Nacional, Centro de Investigaciones Biológicas del Noroeste S.C., México

[2] Cambridge University, Genetics Department, UK

pcarvajal@pg.cibnor.mx, {fborstel, joaquing04, hmejia04, evivas}@cibnor.mx, gr231@cam.ac.uk

Resumen. El contenido de ARN se descifra con fragmentación aleatoria, lo que genera millones de secuencias, que en ausencia de referencias se reconstruyen basándose en algoritmos que usan intensivamente recursos computacionales. Diversos factores afectan el resultado de dicho proceso. Este estudio considera por primera vez cómo la asignación de memoria/núcleos influye sobre la calidad y variabilidad del ensamblaje. Se realizaron múltiples ensamblajes para 2 organismos modelo, en una plataforma monolítica y dos de cómputo de alto desempeño. Se encontraron mayores variabilidades de *contigs* en equipos monolíticos con poca memoria (1.98 y 2.10 veces más que HPC); sin embargo, gran parte de estos (99.16% y 75.79%) mapearon al transcriptoma de referencia demostrando ser de calidad. Por tanto, contrariamente a lo esperado, se observó que una estrategia de ensamblajes múltiples en un equipo de bajos recursos supera el uso de plataformas de alto rendimiento para el descubrimiento de ARNs.

Palabras clave. ARN, secuenciación NGS, RNA-Seq, efecto de memoria en ensamblaje, HPC, optimización de ensamblaje.

Evaluation of the Influence of Computational Resources on Transcriptome *de Novo* Assembly Variability and Quality

Abstract. RNA content is deciphered by random fragmentation of biomolecules, generating millions of sequences. In lack of references these sequences are reconstructed relying on algorithms that require intensive use of computational resources. Numerous factors affect this process. This study explores for the first time how memory/core allocation on reconstruction processes influences assembly quality and variability. Multiple d*e novo* assemblies for two model organisms were obtained from one monolithic platform and two High Performance Computers. Low memory monolithic platforms observed greater variability (1.98 & 2.10 times greater than HPC); however, most of the obtained contigs (99.16% & 75.79%) mapped to the reference transcriptome, thus proving good quality. Therefore, contrary to what was expected, using low-resource equipment when applying assembly strategies that unify numerous assemblies outperforms HPCs on RNA discovery.

Keywords. RNA, NGS sequencing, RNS-Seq, memory effect on assembly, HPC, assembly optimization.

1. Introducción y antecedentes

El genoma de un organismo se puede considerar como el conjunto de instrucciones, codificadas en una secuencia de nucleótidos (ACGT), que contiene toda la información necesaria para formar un organismo y heredar estas características a sus descendientes [1].

La obtención del genoma de un organismo se realiza mediante tecnologías de secuenciación. En particular las tecnologías de secuenciación de

nueva generación (NGS), que generan millones de lecturas cortas y un gran volumen de datos, han revolucionado la biología molecular impactando en áreas como la académica, médica, farmacéutica, biotecnológica, agroquímica y en la industria alimentaria, entre otras [2].

Debido a las limitaciones de los equipos NGS, el primer paso en el proceso de secuenciación consiste en fragmentar la cadena de ácido desoxirribonucleico (ADN) de manera aleatoria, y por ende no se cuenta con información de la posición relativa de cada fragmento [3]. Para poder recuperar la información contenida en la cadena original se requiere entonces un proceso de reconstrucción conocido como ensamblaje.

Existen dos formas primordiales de ensamblaje: la reconstrucción de la cadena alineándola a un genoma de referencia, conocida como mapeo; y la reconstrucción a partir únicamente de los fragmentos, conocida como ensamblaje *de novo*. El ensamblado *de novo* de lecturas de NGS es complejo debido a varios factores, entre ellos, el elevado número de secuencias y su tamaño reducido, así como a los errores de secuenciación y la repetición de cadenas, entre otros [4].

Si bien el conjunto de instrucciones necesarias para el funcionamiento y formación de un organismo se encuentra contenido en su genoma (ADN), las células requieren de moléculas informativas intermediarias, ácidos ribonucleicos (ARN), para dirigir la producción de moléculas de trabajo (proteínas) en un proceso conocido como traducción. El proceso de envío de instrucciones contenidas en el ADN por medio de ARNs se conoce como transcripción, ya que dichas instrucciones no son copias exactas de los segmentos de ADN de las cuales provienen; dichas instrucciones se conocen como transcritos [5]. El término transcriptoma por consiguiente, se refiere al conjunto de ARNs transcritos que representan genes expresándose en un momento dado, proporcionando el estado biológico de la célula [1].

De manera similar a como se realiza la secuenciación de un genoma, se puede secuenciar un transcriptoma.

Pero, un transcriptoma es más complejo; éste contiene miles de transcritos con distinto nivel de abundancia, además una sola secuencia del genoma (gen) puede transcribirse en varias secuencias de transcriptoma (isoformas), debido a que sus fragmentos se pueden alinear de diversas formas (*splicing alternativo*). En consecuencia, cuando se ensamblan las lecturas obtenidas de la secuenciación de un transcriptoma, la fase inicial del ensamblaje realiza una agrupación de las secuencias pertenecientes al mismo gen (*clustering*).

Sin embargo, puede incurrirse en el caso de que los transcritos sean agrupados erróneamente en un mismo *cluster*. Asimismo, otro problema consiste en agrupar genes muy similares (parálogos). Reconstruir todos los transcritos e isoformas que se encuentran expresados, es decir ensamblarlos, ha requerido del desarrollo de nuevos algoritmos computacionales capaces de procesar la gran cantidad de secuencias cortas generadas por las tecnologías NGS [6]. Los algoritmos basados en grafos De Bruijin (DBG) o ensambladores Eulerianos han demostrado ser los más aptos para estas tecnologías [7].

Los algoritmos que usan DBGs se basan en extraer primeramente fragmentos únicos de longitud *k* (*k*-meros) a partir de las secuencias originales; posteriormente se conforman los nodos del grafo con subsecuencias de longitud *k*-1 provenientes de los *k*-meros. A continuación, conectan los nodos considerando prefijos y sufijos (*k*-1-meros) a través de grafos dirigidos. Por último, resuelven la trayectoria a través de un ciclo Euleriano para formar una supercuerda que visite los nodos solo una vez y termine donde empezó [8]. No obstante, el algoritmo presupone condiciones tales como que no existan errores en los nodos, que se encuentre completo el alfabeto de *k*-meros, la existencia de un camino optimo que genere 1 sola supersecuencia y otros que no se cumplen en el proceso de secuenciación. Por ejemplo, la lectura incorrecta de bases puede generar *k*-meros incorrectos, en tanto que la aparición de secuencias repetidas, consecuencia del *splicing alternativo*, implica que más de una supercuerda sea posible [7].

Estos y otros factores relacionados con el proceso biológico, la toma de la muestra, la tecnología de secuenciación, errores de secuenciación, preprocesamiento con base en la calidad de las lecturas obtenidas, la selección del programa para ensamblar e inclusive los

parámetros de ensamblaje influyen en el ensamblaje obtenido. Todos estos factores han sido estudiados en otras investigaciones [9–11]. Sin embargo, un tema que no se ha abordado con suficiente profundidad es la influencia que pudiese tener la disposición y asignación de recursos computacionales para una tarea de ensamblaje *de novo*.

Es común que un proyecto que involucre ensamblaje *de novo* esté enfocado en los aspectos de laboratorio en lugar de los recursos computacionales requeridos. Adicionalmente, una de las principales limitantes para una investigación en la implementación de un ensamblador *de novo* es la demanda computacional de estos programas [12]. Típicamente, un ensamblador *de novo* de vanguardia corre en plataformas multinúcleo, con bastante memoria RAM, con alta capacidad de almacenamiento y sistemas operativos Linux; frecuentemente sobre equipamiento de Cómputo de Alto Rendimiento (*High Performance Computing, HPC*), no disponibles en muchas instituciones.

Los reportes actuales sobre ensamblaje *de novo* típicamente muestran los efectos que la cantidad memoria y núcleos tienen sobre los tiempos de procesamiento o sobre la viabilidad de la plataforma para la ejecución del proceso [12–14]. No obstante, apenas se tienen precedentes de la existencia de una ligera variación en la salida del ensamblaje, resultado de correr el proceso en distintas ocasiones o en hardware distinto [15]. Asimismo, se ha reportado que existe aleatoriedad en los resultados (de ensamblaje) debido al uso de *multi-threading* en combinación con la utilización de estructuras probabilísticas de datos [16].

En ambos trabajos no cuantificaron experimentalmente dicha variabilidad y aleatoriedad en los ensamblajes. Dadas las condiciones típicas de ensamblaje y los pocos precedentes de los efectos computacionales en estos procesos, el efecto que un equipo de cómputo tiene en la obtención de ensamblajes *de novo* no es tomado en cuenta y no se aprovecha en favor de la prospección y generación de información de transcriptomas de especies poco estudiadas.

En este trabajo se considera que la influencia del equipo de cómputo y la asignación de recursos computacionales es relevante en el caso de ensamblaje *de novo* de transcriptoma, dado que el proceso de *clustering* en las primeras etapas a partir de las cuales se determinan las isoformas, depende de la disponibilidad inicial de secuencias y la incorporación sucesiva de secuencias candidatas al *cluster*. La distribución inicial de secuencias en las localidades de memoria disponibles para cada procesador influirá, por ende, en la formación de estos *clusters* y en el resultado final del ensamblaje.

A partir de esta hipótesis, en este trabajo se explora la influencia que tiene la asignación de los recursos computacionales en el ensamblaje *de novo* de transcriptoma, en términos de repetitividad de un ensamble bajo las mismas condiciones, entre condiciones distintas, y en términos de calidad al comparar lo obtenido con un transcriptoma de referencia.

2. Metodología

Para evaluar la repetitividad y calidad se realizaron ensamblajes en diferentes plataformas computacionales, utilizando organismos modelo para los cuales, sí existe una referencia contra la cual comparar y determinar la calidad de los transcriptomas ensamblados. Se decidió utilizar el software ensamblador de transcriptomas Trinity (ver. 2.1.1) [17], por ser considerado por la comunidad científica como el ensamblador por defecto para realizar ensamblaje *de novo* [13].

2.1. Recursos computacionales

Dado que el ensamblaje *de novo* de transcriptoma demanda el uso de cómputo intensivo, se seleccionó como sistema mínimo una estación de trabajo y dos sistemas de HPC, variando en cada plataforma la asignación de memoria y de núcleos de cómputo, tal como se muestra en la Tabla 1. Todas las plataformas cuentan con sistemas operativos Linux de 64 bits.

En la estación de trabajo la memoria RAM se limitó físicamente, en tanto que en las plataformas de HPC, dado que no es posible modificar físicamente sus recursos de hardware, se variaron los recursos asignados al trabajo de ensamblaje a través del planificador correspondiente, en cuanto

Tabla 1. Configuraciones de las plataformas de cómputo para realizar ensamblajes

Plataforma de cómputo				Parámetros Trinity		Planificador	
						SLURM	TORQUE
Nombre	Plataforma	Memoria RAM (GB)	Núcleos	Máxima Memoria	CPU	Núcleos	Nodo/ Núcleos
W_1	Estación de trabajo	20	6	20	6	-	-
W_2		24	6	24	6	-	-
H_1	HPC (No Virtual)	128/nodo	20/nodo	24	6	-	1/6
H_2		128/nodo	20/nodo	64	10	-	1/10
V_1	HPC (Virtual)	128/nodo	24/nodo	24	6	6	-
V_2		128/nodo	24/nodo	64	12	12	-

"-" equivale a *"No aplica"*

al número de nodos de cómputo y núcleos por nodo.

En el ensamblador Trinity se asignó el parámetro de memoria y el número de hilos por CPU para corresponder con las configuraciones de hardware. Los demás parámetros fueron especificados a sus valores por defecto.

Se utilizó una estación de trabajo *Dell Precision* T7500, con un procesador Intel Xeon X5680 3.3 GHz de 6 núcleos, con capacidad en discos duros de 2.5 TB, a la cual se le modificó la memoria RAM (20 GB, 24 GB) de acuerdo a las configuraciones W_1 *y* W_2, respectivamente (Tabla 1).

Los centros HPC donde se realizaron los procesos de ensamblaje son: Laboratorio Nacional de Supercómputo del Sureste de México (LNS) de la Benemérita Universidad Autónoma de Puebla [18] y el proveedor *Penguin Computing*, a través de su servicio en la nube *Penguin On Demand* [19].

El primer recurso de HPC está conformado por la supercomputadora Cuetlaxcoapan del LNS, compuesta de un *cluster* estándar de cálculo con procesadores Intel Xeon y un *cluster* con procesadores Intel Xeon Phi Knights Landing. El *cluster* estándar está compuesto de 228 nodos de cálculo *Thin* y otros 42 nodos de cálculo más robustos (*fat, semi-fat, ultra-fat*). Para los procesos de ensamble realizados en este estudio se utilizó el *cluster* estándar, el cual funciona en modo virtual y donde los nodos de cálculo *Thin*, tienen 2 procesadores Intel Xeon E5-2680 v3 (Haswell) a 2.5 GHz, con 24 núcleos en total y 128 GB de memoria RAM. Los nodos están intercomunicados con una red Ethernet Gigabit y una red Infiniband FDR a 56 Gbps. Este *cluster* utiliza el administrador de carga de trabajos SLURM [20], que es libre y puede manejar un *cluster* Linux de cualquier dimensión. La especificación de los recursos computacionales a utilizar en cada ensamblaje se definió en el Job Script, a través de los parámetros de SLURM:

#SBATCH -n 24 # number of MPI tasks (cores) requested,

#SBATCH --ntasks-per-node=24 # task (cores) per node (maximum 24).

Este ejemplo especifica que se ejecute el trabajo con 24 núcleos (1 nodo *Thin*) del *cluster* estándar, donde cada núcleo obtiene 5.3 GB de RAM [18]. Esta plataforma computacional, se utilizó para realizar los ensambles en las configuraciones V_1 y V_2 (Tabla 1), especificando el uso de 1 nodo en el *Job Script*, pero variando la cantidad de núcleos en concordancia con los parámetros de entrada de Trinity.

El segundo recurso de HPC utilizó el servicio en la nube POD de *Penguin Computing* con la cola T30, que especifica nodos con un procesador Intel Xeon E5-2660 v3 (Haswell) a 2.6 GHz con 20 núcleos y 128 GB de RAM.

Todos los nodos están intercomunicados con una red Ethernet Gigabit de 10 Gbps y una red

Infiniband QDR a 40 Gbps. El servicio utiliza el planificador PBS TORQUE [21] para introducir trabajos al *cluster* computacional. Sin embargo, es necesario seleccionar una cola del planificador. Cada cola provee diferentes tipos de nodo de cómputo, y por lo tanto tienen diferentes precios.

La especificación de los recursos se definió en el *Job Script*, a través de los parámetros de TORQUE:

#PBS -q T30,

#PBS -l nodes=1:ppn=20.

Este ejemplo especifica que se ejecute el trabajo con 1 nodo de 20 núcleos de la cola T30, donde cada núcleo tiene 6.4 GB de RAM [19]. Se utilizó esta plataforma computacional, especificando en el *Job Script* el uso de 1 nodo, pero variando el número de núcleos en concordancia con los parámetros de entrada de Trinity; estas disposiciones se utilizaron para realizar los ensambles en las configuraciones H_1 *y* H_2 de la Tabla 1.

2.2. Monitoreo de memoria

Con la finalidad de cuantificar el uso de recursos durante el proceso de ensamblaje se decidió monitorear el proceso mediante el comando *top* de Linux. Posteriormente, se analizaron los datos con Matlab (r2013a) [22]. El registro generado a partir de un muestreo cada 10 segundos y el registro de tiempos (*Trinity.timing*) permiten identificar la etapa del proceso de ensamblaje que hace el uso más extensivo de memoria.

2.3. Organismos

Aun cuando el ensamblaje *de novo* se utiliza principalmente en organismos que no cuentan con un genoma o transcriptoma de referencia, en este estudio se requieren referencias para obtener la calidad de los ensamblajes, por lo que se seleccionaron dos organismos modelo: la Mosca de la Fruta (*Drosophila melanogaster*) y la Pulga de Agua (*Daphnia pulex*).

Las lecturas crudas de la Mosca de la Fruta, fueron obtenidas del Sequence Read Archive (SRA) [23] del Centro Nacional de Información Biotecnológica (NCBI), con número de identificación SRR042489, provenientes del proyecto [24]. Las lecturas de la Pulga de Agua fueron descargadas de repositorio ENA [25] perteneciente al Instituto Europeo de Bioinformática, número de identificación SRR2075894, obtenidas en el proyecto [26].

Se analizó la calidad de las lecturas con FastQC [27] y según sus resultados se pre-procesaron los datos con Trimmomatic (versión 0.32) [28].

Los transcriptomas de referencia fueron descargados del repositorio Ensembl [29]. Para la Mosca de la Fruta fue la versión 87, que contiene 30,651 transcritos. Para la Pulga de Agua fue la versión GCA_000187875.1 con 30,590 transcritos.

2.4. Métricas

2.4.1. Repetitividad por plataforma computacional

Dado un conjunto de lecturas de secuenciación L_m de una especie m obtenidas de una base de datos pública, sea $E_{(p,m,i)}$ un ensamble (un conjunto constituido por *contigs*) realizado con configuración de plataforma computacional p, utilizando L_m como entrada al proceso de ensamblaje *de novo*, donde i es el número correspondiente a la repetición del proceso de ensamblaje utilizando las mismas condiciones iniciales, se tiene que:

$$I_{(p,m,n)} = \bigcap_{i=1}^{i=n} E_{(p,m,i)}, \quad (1)$$

donde $I_{(p,m,n)}$ representa el conjunto de *contigs* resultantes de la intersección entre los n ensambles *de novo*, con las mismas condiciones.

Asimismo, se puede decir que el conjunto de *contigs* no intersectados $\bar{I}_{(p,m,n)}$ es tal, que:

$$\bar{I}_{(p,m,n)} \cap I_{(p,m,n)} = \{\}. \quad (2)$$

Por lo tanto, el conjunto $\bar{I}_{(p,m,n)}$ representa el conjunto de *contigs* que no aparecen en todos los ensambles $E_{(p,m,i)}$ que se generaron durante algún ensamblaje en particular.

De tal manera que la cantidad total de *contigs* obtenidos por plataforma p para un organismo m

en n repeticiones está dada por la unión de sus conjuntos intersectados y no intersectados:

$$C\_total_{(p,m,n)} = I_{(p,m,n)} \cup \bar{I}_{(p,m,n)}. \quad (3)$$

Los análisis de conjuntos intersectados y no intersectados se realizaron con el software de cómputo científico Matlab (r2013a) [22].

La cuantificación de la repetitividad se da con base al porcentaje que representa el subconjunto $I_{(p,m,n)}$ del conjunto $Ctotal_{(p,m,n)}$ y la variabilidad se da con base al porcentaje que representa el subconjunto $\bar{I}_{(p,m,n)}$ del conjunto $Ctotal_{(p,m,n)}$, encontrados por plataforma p para el organismo m en n repeticiones.

Finalmente, la ganancia por variabilidad entre plataformas se cuantifica tomando en cuenta la relación de la variabilidad máxima de las configuraciones de la estación de trabajo entre la variabilidad máxima de las configuraciones en las plataformas basadas en HPC.

2.4.2. Calidad

Si bien existe variabilidad entre un ensamblaje y otro, aun partiendo de las mismas condiciones iniciales, es necesario considerar si esta variabilidad representa *contigs* presentes en el transcriptoma del organismo o son artefactos del proceso matemático-computacional. Para identificar la validez de los *contigs* generados es necesario realizar un proceso de mapeo de los *contigs* al transcriptoma de referencia, definiendo así la calidad del ensamblaje.

El término calidad de ensamblaje, se refiere a la concordancia entre el ensamblaje y el transcriptoma original del organismo modelo [30] o en este caso, a su referencia codificante más próxima y el conjunto de *contigs* obtenidos del proceso de ensamblado. En este estudio el objetivo es detectar la calidad de los contigs originados exclusivamente por la variabilidad de cada plataforma de cómputo.

En este contexto, se define como referencia codificante a la mejor aproximación de ensamblado de transcriptoma disponible en cierta fecha o versión, disponible en las bases públicas de los organismos involucrados, ver el sitio de Ensembl [13, 29].

Formalmente, sean los conjuntos $I_{(p,m,n)}$ e $\bar{I}_{(p,m,n)}$ se procede a analizarlos con respecto al conjunto referencia codificante de la especie, llamado Transcriptoma de referencia T_m, mediante un proceso de identidad, utilizando el software BLAST [31]; de tal manera que:

$$\{(c,t): c \in Im_{(p,m)}, t \in T_{(p,m)}\} \; = f : I_{(p,m)} \to T_m \,, \quad (4)$$

donde (c,t) representa un par (*contig*, transcrito) y $Im_{(p,m)}$ es el conjunto de *contigs* intersectados generados con la plataforma computacional p, que mapearon en el conjunto referencia T_m. $T_{(p,m)}$ es el subconjunto de transcritos de T_m a los que mapearon los *contigs* intersectados del conjunto $Im_{(p,m)}$. Nótese que, para simplificar, se omitió el subíndice n, así mismo en:

$$\{(c,t): c \in \bar{I}m_{(p,m)}, t \in \bar{T}_{(p,m)}\} \; = f : \bar{I}_{(p,m)} \to T_m \,, \quad (5)$$

donde (c,t) representa un par (*contig*, transcrito) e $\bar{I}m_{(p,m)}$ es el conjunto de *contigs* no intersectados generados con la plataforma computacional p, que mapearon en el conjunto referencia T_m. $\bar{T}_{(p,m)}$ es el subconjunto de transcritos de T_m a los que mapearon los *contigs* no intersectados del conjunto $\bar{I}m_{(p,m)}$.

Ya que pueden existir *contigs* intersectados y no intersectados que mapean a un transcrito común, se puede realizar la siguiente operación:

$$T_{(p,m)} \cap \bar{T}_{(p,m)} = \; T^*_{(p,m)} \,, \quad (6)$$

donde $T^*_{(p,m)}$ es el subconjunto de transcritos mapeados compartidos por los conjuntos de *contigs* intersectados y no intersectados. Para obtener los *contigs* no intersectados compartidos, se realiza:

$$\{(c,t): c \in \bar{I}m^*_{(p,m)}, t \in T^*_{(p,m)}\} = f : \bar{I}_{(p,m)} \to T^*_{(p,m)} \,, \quad (7)$$

donde $\bar{I}m^*_{(p,m)}$ es el conjunto de *contigs* mapeados no intersectados compartidos.

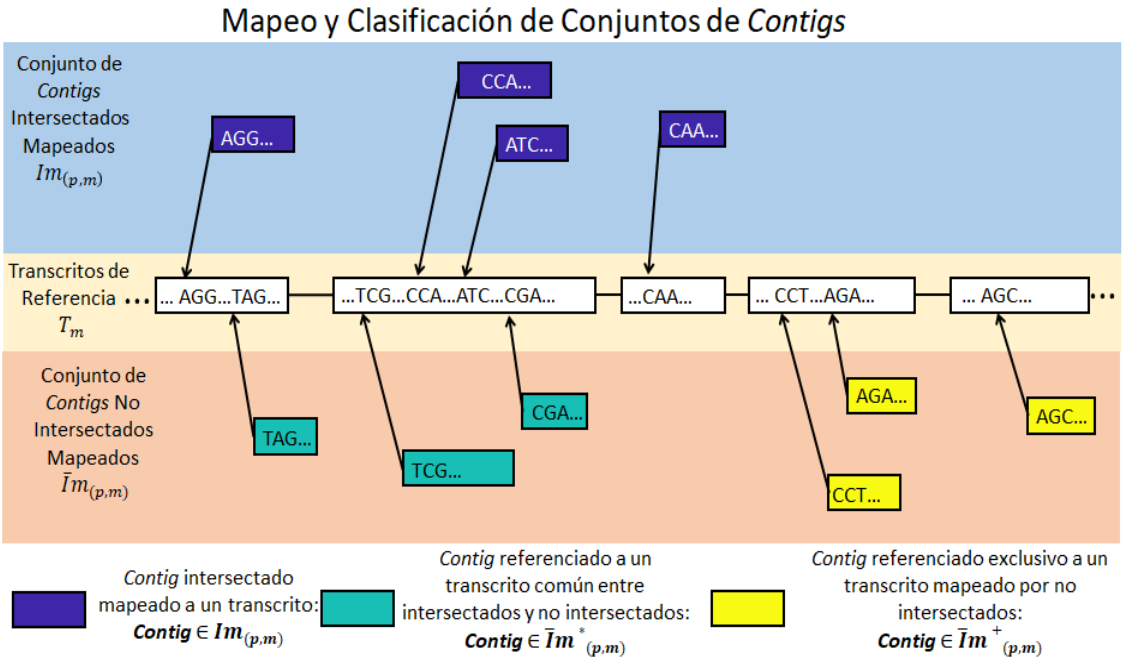

Fig. 1. Mapeo de *contigs* intersectados y no intersectados, así como su clasificación dependiendo del transcripto al que hayan sido referenciados

Asimismo, se puede realizar la siguiente operación:

$$\bar{I}m_{(p,m)} - \bar{I}m^{*}_{(p,m)} = \bar{I}m^{+}_{(p,m)}, \tag{8}$$

donde $\bar{I}m^{+}_{(p,m)}$ es el conjunto de *contigs* no intersectados que exclusivamente mapean a transcritos en T_m que no son compartidos con el conjunto de transcritos mapeados inicialmente por $Im_{(p,m)}$. La representación de estos conjuntos con respecto al transcriptoma de referencia se muestra en la Figura 1.

Para decidir que *contigs* se compartían entre ensamblajes se utilizó un criterio estricto de coincidencia única, es decir, ambas cadenas deberían ser exactamente iguales en tamaño y composición. No obstante, los algoritmos de mapeo tienen un criterio más laxo, permitiendo reconocer secciones similares aun cuando no sean las cadenas exactamente iguales. Este criterio permite variar la longitud y composición del *contig*.

Debido a este criterio pudiese haber *contigs* que siendo ligeramente distintos mapean a la misma porción del transcriptoma.

Esta situación puede ocurrir para ambos conjuntos (comunes y no compartidos) y entre conjuntos. Por ello, la evaluación de calidad considera como información válida originada por variabilidad de plataforma a todos aquellos *contigs* contenidos en el subconjunto $\bar{I}_{(p,m,n)}$. Esta evaluación se expresa en el porcentaje representado por los *contigs* mapeados provenientes del conjunto no intersectado $\bar{I}m_{(p,m)}$, por plataforma p para un organismo m con respecto al $\bar{I}_{(p,m,n)}$, dados n ensambles.

Asimismo, considera como información nueva originada por la variabilidad de plataforma solo a los *contigs* mapeados exclusivos al conjunto no intersectado $\bar{I}m^{+}_{(p,m)}$. Esta evaluación se expresa en el porcentaje representado por los *contigs* mapeados exclusivos provenientes del conjunto no intersectado $\bar{I}m^{+}_{(p,m)}$, por plataforma p para un organismo m con respecto al $\bar{I}_{(p,m,n)}$, dados n ensambles.

Tabla 2. Cantidad de *Contigs* ensamblados por plataforma computacional

	Mosca de la Fruta		Pulga de Agua	
Plataforma Computacional	**Promedio**	**Desviación Estándar**	**Promedio**	**Desviación Estándar**
$\boldsymbol{W_1}$	25,994.80	6.72	53,280.4	37.63
$\boldsymbol{W_2}$	25,988.80	5.81	53,276.8	39.93
$\boldsymbol{H_1}$	25,981.60	3.44	53,220.8	20.32
$\boldsymbol{H_2}$	25,984.40	5.68	53,247.8	15.08
$\boldsymbol{V_1}$	25,988.40	3.65	53,321.6	23.06
$\boldsymbol{V_2}$	25,989.40	2.30	53,320.0	23.98

La ganancia en calidad se cuantifica tomando en cuenta la relación del número máximo de *contigs* representados por el conjunto $\bar{I}m_{(p,m)}$ de una de las configuraciones basadas en la estación de trabajo, entre el número máximo de *contigs* representados en $\bar{I}m_{(p,m)}$ de una de las configuraciones basadas en HPC.

De la misma forma, la ganancia en información nueva se cuantifica tomando en cuenta la relación del número máximo de *contigs* representados por el conjunto $\bar{I}m^{+}_{(p,m)}$de una de las configuraciones basadas en la estación de trabajo, entre el número máximo de *contigs* representados en $\bar{I}m_{(p,m)}$ de una de las configuraciones basadas en HPC.

Cabe hacer notar que los parámetros del software BLAST fueron establecidos de tal manera que se obtuviese un *hit* (alineamiento positivo) por secuencia de entrada, alta similitud entre secuencias alineadas, pero con bajos valores de expectación. Umbrales utilizados: valor de expectación *e-value* $1x10^{-9}$; porcentaje de identidad: 95%; alineamientos máximos por secuencia de entrada *max_hps*: 1; cantidad de secuencias alineadas *max_target_seqs*:1; cantidad de núcleos: 1.

3. Resultados

3.1. Datos de entrada

Las estrategias de pre procesamiento para los datos de secuenciación se realizaron basadas en los correspondientes reportes de calidad de las lecturas para cada organismo. Para Mosca de la Fruta fue: corte de primeras 10 bases, remoción de adaptadores en modo palíndromo, $l > 32$. En los datos de Pulga de Agua: corte de primeras 10 bases, remoción de adaptadores en modo palíndromo, Q_{min} 25, $l > 32$; posterior al preprocesamiento los datos seguían conteniendo secuencias sobrerrepresentadas, ribosómicas según la búsqueda en la base de datos del NCBI [32], consecuentemente se realizó un segundo pre-procesamiento para remover dichas secuencias.

Los datos de entrada pre-procesados fueron 7,564,138 y 7,168,393 lecturas pareadas con longitudes variables de 32-66 bases para Mosca de la Fruta y Pulga de Agua, respectivamente.

3.2 *Contigs* ensamblados

La Tabla 2 muestra el promedio de *contigs* generados después de 5 repeticiones del ensamblaje efectuados en cada configuración. De los promedios y desviaciones se puede apreciar que el número de *contigs* generados en cada repetición es muy parecido.

3.3. Uso de memoria

El ensamblador Trinity consta de tres módulos: *Inchworm*, donde realiza la agrupación inicial de lecturas pertenecientes al mismo gen (*clustering*) y una construcción extendida de secuencias con base en dichos *clusters*; *Chrysalis*:

Tabla 3. Utilización máxima de memoria RAM (GB) por plataforma computacional, organismo y módulo de ensamblaje de Trinity

	Mosca de la Fruta		Pulga de Agua	
Plataforma Computacional	**Inchworm**	**Chr/Btf**	**Inchworm**	**Chr/Btf**
W_1	20.4	9.7	20.4	20.4
W_2	24.4	12.8	24.5	24.5
H_1	21.1	13.0	26.0	27.8
H_2	78.3	23.3	75.7	34.3

Inchworm: Primer módulo del ensamblador. Chr/Btf: Segundo y tercer módulo del ensamblador Trinity, *Chrysalis y Butterfly* respectivamente

- Es el módulo donde construye los grafos De Bruijn con base en los *clusters* de lecturas y los *contigs* extendidos, para finalmente pasar al módulo; *Butterfly*, donde resuelve ambigüedades en los grafos con base en la cantidad de lecturas que respaldan la trayectoria de análisis [15].

El módulo *Inchworm* es el primero en ser ejecutado. Posteriormente, los módulos *Chrysalys y Butterfly* se ejecutan de forma alterna.

La Figura 2 muestra el uso de memoria RAM durante los procesos de ensamblaje *de novo* de transcriptoma para Mosca de la Fruta y Pulga de Agua en las plataformas H_2 y W_2. Se delimitó con una línea vertical intermitente la duración del módulo *Inchworm*.

Nótese que el manejo de los millones de secuencias de entrada se ven reflejadas en el manejo de memoria por parte del ensamblador, sobre todo en el primer módulo, *Inchworm*, donde ambos organismos hicieron uso intensivo de memoria, ~80 GB en la plataforma H_2 (Figura 2 a y b). La duración del primer módulo en las configuraciones H_2 fue de ~24 minutos en procesos de Mosca de la Fruta y ~5 minutos en ensamblajes de Pulga de Agua. Asimismo, se puede observar que el uso de memoria en los módulos posteriores fue mayor en los procesos de Pulga de Agua teniendo un pico de uso en 34.3 GB (Tabla 3). El uso de memoria en los módulos alternados *Chrysalis* y *Butterfly* en el caso de Mosca de la Fruta no excedió los 24 GB.

El uso de memoria en las plataformas W_1 y W_2 se vio limitado por la capacidad física de memoria de las plataformas, inclusive los ensamblajes de Mosca de la Fruta tendieron a saturar el primer módulo (Figura 2 c), y se observaron picos máximos de memoria en al menos 2 módulos al procesar la Pulga de Agua (Figura 2 d).

En la Tabla 3 se muestra la utilización máxima de memoria por configuración, especie y módulo de Trinity.

3.4. Repetitividad por intersecciones

Las figuras 3 y 4 muestran el resultado de intersectar los conjuntos generados con las 5 repeticiones del ensamblaje por máquina. Las intersecciones son los *contigs* comunes a los ensamblajes que contienen exactamente las mismas secuencias. Basta un cambio de base, inserción o perdida entre dos *contigs* para que se consideren ambos distintos y se envíen al conjunto de no intersectados. Nótese la diferencia de escalas, y que los *contigs* no intersectados constituyen menos del 5% en el caso de la Mosca de la Fruta y menos del 30% para Pulga de Agua.

En la Tabla 4 se muestran los porcentajes de repetividad y variabilidad por plataforma para ambos organismos, encontrándose mayor repetitividad en las plataformas HPC, pero mayor variabilidad en las plataformas con menor memoria. Se observa también que la ganancia máxima por variabilidad de una de las configuraciones basadas en la estación de trabajo, para Mosca de la Fruta es 4.49/2.26 = 1.98 veces

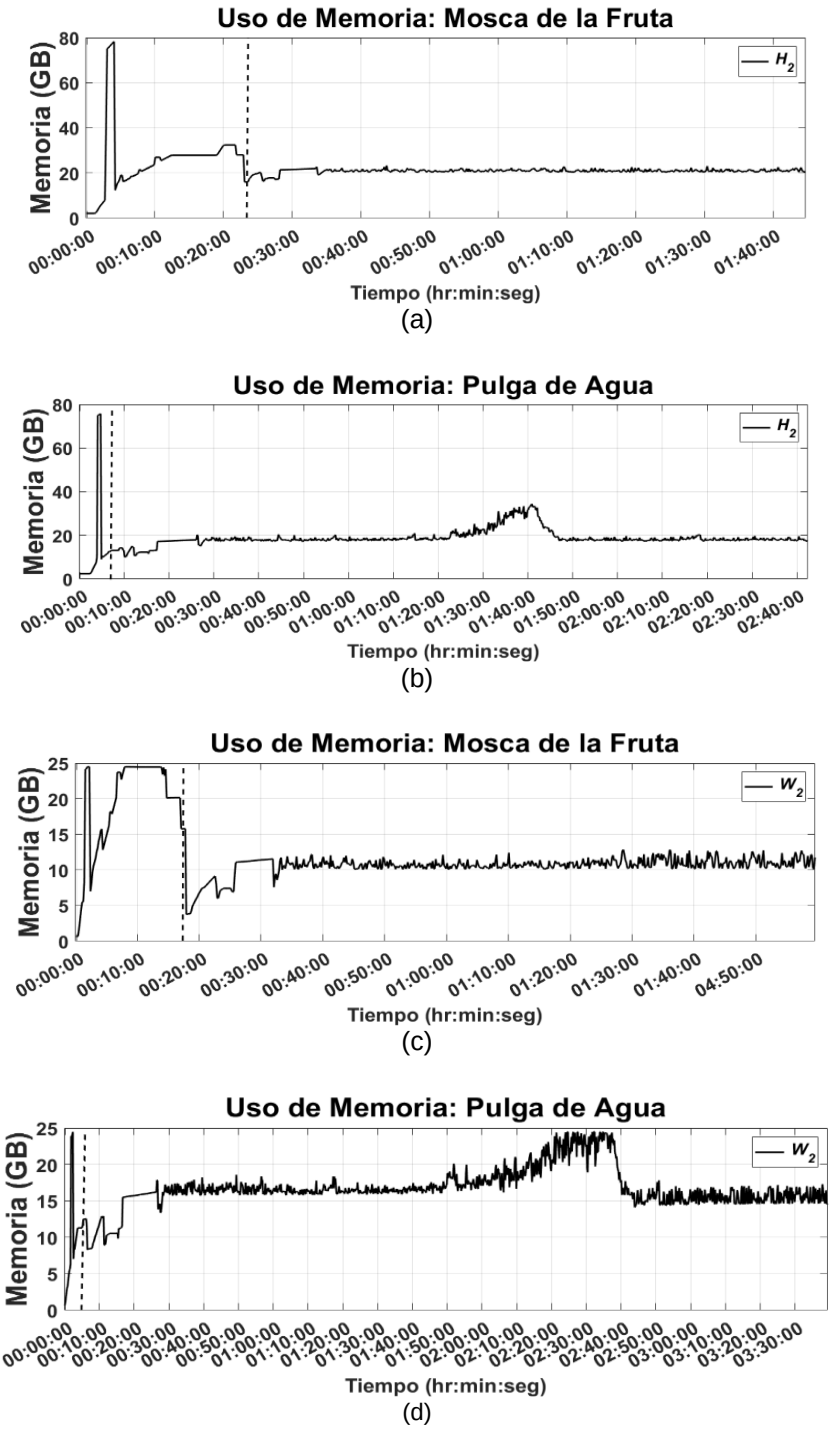

Fig. 2. Utilización de memoria RAM durante el ensamblaje en la plataforma H_2 de Mosca de la Fruta en las configuraciones (a) y Pulga de Agua (b); ensamblaje en la plataforma W_2 de Mosca de la Fruta (c) y Pulga de Agua (d). Línea vertical intermitente indica fin del módulo *Inchworm*

Tabla 4. Repetitividad y Variabilidad

Mosca de la Fruta					
Plataforma Computacional	$Ctotal_{(p,mosca,5)}$	$I_{(p,mosca,5)}$	**Repetitividad (%)**	$\bar{I}_{(p,mosca,5)}$	**Variabilidad (%)**
W_1	26,618	25,422	95.51	1,196	4.49
W_2	26,544	25,474	95.97	1,070	4.03
H_1	26,286	25,693	97.74	593	2.26
H_2	26,286	25,692	97.74	594	2.26
V_1	26,202	25,784	98.40	418	1.60
V_2	26,180	25,807	98.58	373	1.42
Pulga de Agua					
Plataforma Computacional	$Ctotal_{(p,pulga,5)}$	$I_{(p,pulga,5)}$	**Repetitividad (%)**	$\bar{I}_{(p,pulga,5)}$	**Variabilidad (%)**
W_1	60,632	47,510	78.36	13,122	21.64
W_2	60,943	47,261	77.55	13,682	22.45
H_1	56,617	50,590	89.35	6,027	10.65
H_2	56,630	50,580	89.32	6,050	10.68
V_1	55,545	51,627	92.95	3,918	7.05
V_2	55,176	51,783	93.85	3,393	6.15

Repetitividad: $I_{(p,m,n)}$ / $Ctotal_{(p,m,n)}$. Variabilidad: $\bar{I}_{(p,m,n)}$ / $Ctotal_{(p,m,n)}$. Máxima Variabilidad entre plataformas W_1 y W_2, y máxima variabilidad entre plataformas HPC remarcadas en gris

más que el máximo de una de las plataformas basadas en HPC; para Pulga de Agua es 22.45/10.68 = 2.10 veces más que el máximo de una de las plataformas basadas en HPC.

3.5. Calidad por mapeos

De igual forma, el mapeo de los *contigs* de los conjuntos no intersectados por plataforma fue mayor para los conjuntos provenientes de las plataformas con menor memoria (W_1 y W_2).

Las figuras 5 y 6 muestran el mapeo por plataforma computacional para los organismos Mosca de la Fruta y la Pulga de Agua, respectivamente. También, se observan en la Tabla 5 los mapeos de *contigs* intersectados con respecto a las referencias y los porcentajes que estos representan.

La ganancia máxima de *contigs* no intersectados mapeados, para Mosca de la Fruta es 1,186/584 = 2.03 veces más que el máximo de una de las plataformas basadas en HPC; asimismo, para Pulga de Agua es 10,057/4,327 = 2.39 veces más.

La ganancia máxima de *contigs* no intersectados mapeados exclusivos, para Mosca de la Fruta es 315/182 = 1.73 veces más que el máximo porcentaje de las plataformas basadas en HPC; asimismo, para Pulga de Agua es 4,017/1,628 = 2.46 veces más.

4. Discusión

La aproximación inicial de los recursos mínimos computacionales para ensamblaje *de novo* está dada por los requerimientos básicos del software ensamblador y la cantidad de datos de entrada. El requerimiento mínimo de memoria para ensamblaje *de novo* de transcriptoma reportado en el ensamblador Trinity es ~ 1GB de memoria por

ISSN 2007-9737

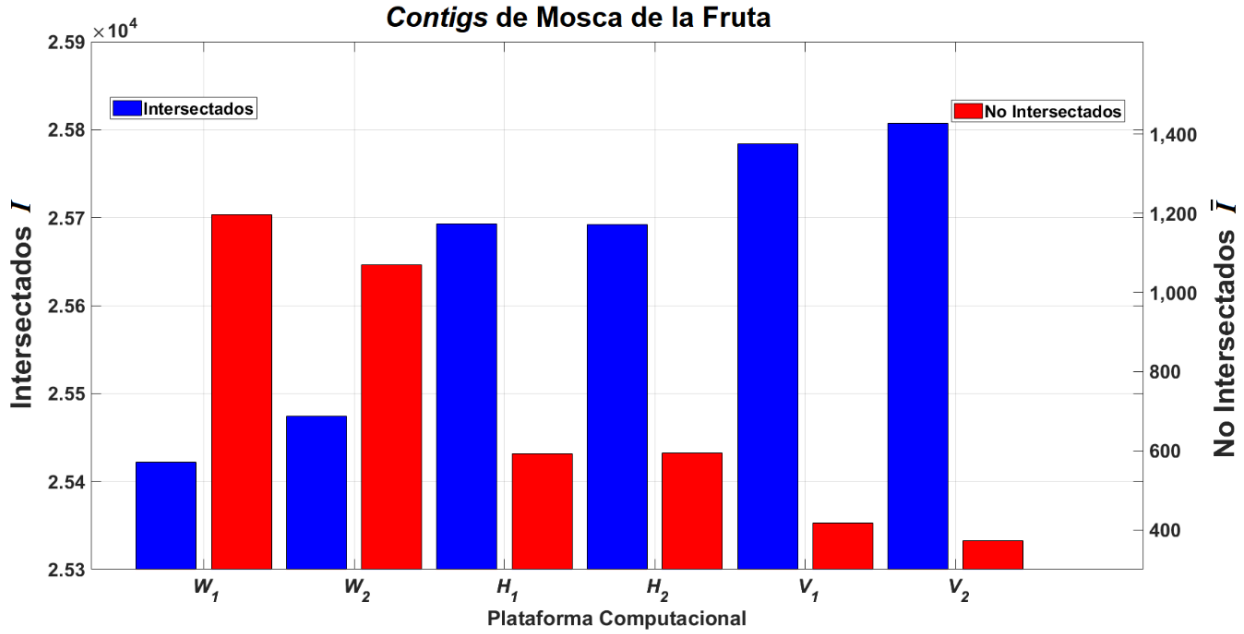

Fig. 3. Comparación de los conjuntos de *contigs* intersectados $I_{(p,mosca,5)}$ y los conjuntos de *contigs* no intersectados $\bar{I}_{(p,mosca,5)}$ obtenidos después de 5 repeticiones de ensamblajes $E_{(p,mosca,5)}$ para el organismo Mosca de la Fruta, por cada plataforma computacional p de la Tabla 1

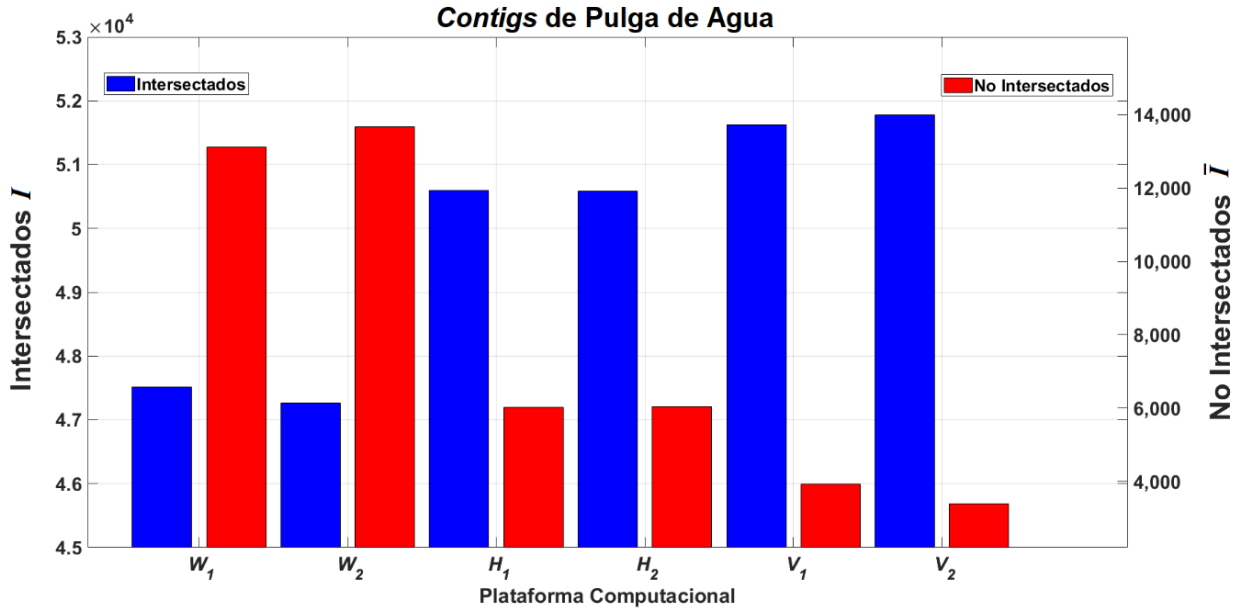

Fig. 4. Comparación de los conjuntos de *contigs* intersectados $I_{(p,pulga,5)}$ y los conjuntos de *contigs* no intersectados $\bar{I}_{(p,pulga,5)}$, obtenidos después de 5 repeticiones de ensamblajes $E_{(p,pulga,5)}$ para el organismo Pulga de Agua, por cada plataforma computacional p de la Tabla 1

Tabla 5. Mapeos de *contigs* con respecto a las referencias

Mosca de la Fruta					
Plataforma Computacional	$\bar{Im}_{(p,mosca)}$	$\bar{Im}^{+}_{(p,mosca)}$	$\bar{I}_{(p,mosca,5)}$	$\bar{Im}_{(p,mosca)}/\bar{I}_{(p,mosca,5)}$ (%)	$\bar{Im}^{+}_{(p,mosca)}/\bar{I}_{(p,mosca,5)}$ (%)
W_1	1,186	315	1,196	99.16	26.33
W_2	1,054	312	1,070	98.50	29.15
H_1	584	160	593	98.48	26.93
H_2	583	182	594	98.15	30.63
V_1	409	87	418	97.85	20.81
V_2	364	86	373	97.59	23.05
Pulga de Agua					
Plataforma Computacional	$\bar{Im}_{(p,pulga)}$	$\bar{Im}^{+}_{(p,pulga)}$	$\bar{I}_{(p,pulga,5)}$	$\bar{Im}_{(p,pulga)}/\bar{I}_{(p,pulga,5)}$ (%)	$\bar{Im}^{+}_{(p,pulga)}/\bar{I}_{(p,pulga,5)}$ (%)
W_1	10,057	3,831	13,122	76.64	29.19
W_2	10,369	4,017	13,682	75.79	29.35
H_1	4,245	1,628	6,027	70.43	27.01
H_2	4,327	1,567	6,050	71.52	25.90
V_1	2,385	1,323	3,918	60.87	33.76
V_2	2,074	1,119	3,393	61.13	32.97

$\bar{Im}_{(p,m)}/\bar{I}_{(p,m,n)}$ y $\bar{Im}^{+}_{(p,m)}/\bar{I}_{(p,m,n)}$ expresado en porcentaje.
Máximos entre plataformas W_1, W_2 y en HPC remarcados en gris.

cada millón de lecturas de entrada [17]. Según esta estimación, para los datos de Mosca de la Fruta (~7.5 millones de lecturas) y Pulga de Agua La memoria en todas las plataformas era mucho mayor que los requerimientos mínimos teóricos para ensamblaje, para los datos en ambos organismos, como se indicó en la Tabla 1. Con base en los requerimientos mínimos teóricos de ensamblaje, la asignación de memoria o memoria disponible por plataforma fue calculada a más del doble o el triple.

En la práctica los requerimientos mínimos reales de memoria para ensamblar excedieron el límite teórico, utilizando más del triple de este; tal como se observa en la Figura 2 (a y b), donde se graficó la utilización de memoria en H_2, y asimismo en la Figura 2 (c y d) para W_2, al ensamblar ambos organismos. El ensamblador tiende a utilizar cuanta memoria esté disponible para realizar sus procesos, independientemente del parámetro de uso de memoria asignado.

Se ha reportado que el primer módulo de Trinity es más extensivo en el uso de memoria [15], lo cual coincide con las gráficas de la Figura 2. Según la asignación de memoria para H_2 el límite de uso debió haber sido muy cercano a 64 GB, pero éste fue excedido por más de 11 GB en ambos organismos (ver Tabla 3).

Caben destacar que los ensambles generados por las plataformas V_1 y H_1, en donde el uso de memoria, asignado en Trinity, estaba limitado a la misma cantidad de GB que en la plataforma W_2, y cuyos conjuntos intersecciones de *contigs* fueron mayores, tuvieron mayor disponibilidad de memoria, ya que como se demostró con anterioridad, el parámetro de uso de memoria del ensamblador no fue una limitante para la utilización del recurso.

De esta manera, se puede decir que los análisis aquí presentados fueron realizados en plataformas con tres cantidades distintas de memoria 20, 24 y 128 GB, estando representadas

ISSN 2007-9737

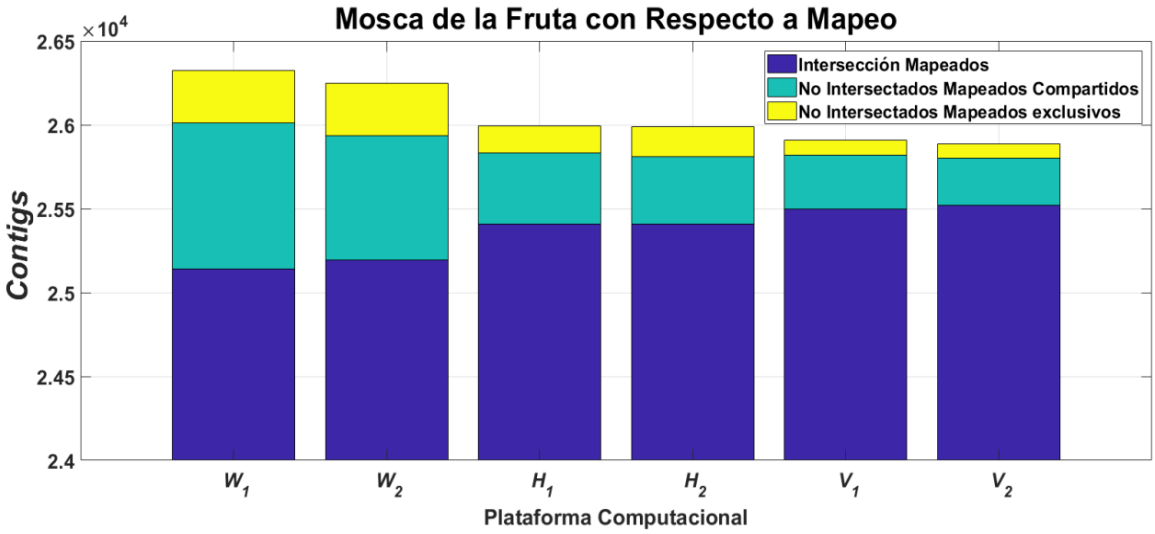

Fig. 5 Comparación de los conjuntos de *contigs* intersectados mapeados $Im_{(p,mosca)}$ al transcriptoma de referencia de la Mosca de la Fruta T_{mosca}, *contigs* no intersectados mapeados exclusivos $\bar{I}m^{+}_{(p,mosca)}$, y *contigs* no intersectados compartidos $\bar{I}m^{*}_{(p,mosca)}$, haciendo 5 ensamblajes $E_{(p,mosca,5)}$, por cada plataforma computacional p. Nótese que la escala inicia en 2.4x10^4

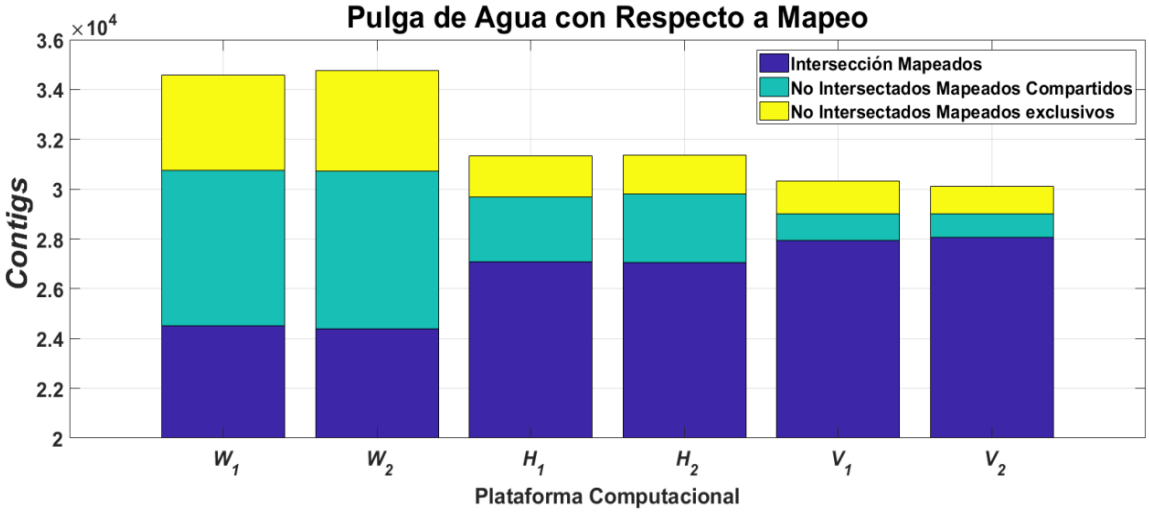

Fig. 6 Comparación de los conjuntos de *contigs* intersectados mapeados $Im_{(p,pulga)}$ al transcriptoma de referencia de la Mosca de la Fruta T_{pulga}, *contigs* no intersectados mapeados exclusivos $\bar{I}m^{+}_{(p,pulga)}$ y *contigs* no intersectados compartidos $\bar{I}m^{*}_{(p,pulga)}$, haciendo 5 ensamblajes $E_{(p,pulga,5)}$, por cada plataforma computacional p. Nótese que la escala inicia en 2x10^4

las primeras dos en las plataformas de menor memoria (W_1 y W_2, respectivamente) y la última en las de mayor memoria (V_1, V_2, H_1, *y* H_2), en donde la configuración de memoria por parámetro no fue tomada en cuenta al momento de ejecutar los ensamblajes.

De este modo se empiezan a observar los efectos de las plataformas de cómputo sobre el

conjunto de *contigs* de ensamblajes debido a la disponibilidad de memoria física.

Se observa en la Tabla 2 que las desviaciones estándar indican muy poca variación en la cantidad de *contigs* construidos, de 2.3 a 6.7 para Mosca de la Fruta y de 15.08 a 39.9 para Pulga de Agua, tendiendo a ser mayores para las plataformas de menor memoria (W_1 y W_2). Si se dejase en este punto la exploración de resultados se encontraría que estas variaciones son muy pequeñas en comparación con la cantidad de *contigs* obtenidos. Sin embargo, el efecto que los recursos computacionales tuvieron sobre el conjunto de *contigs* es encontrado en el análisis del contenido de éstos.

En ambos organismos se encontró mayor repetitividad en los ensamblajes procesados por las plataformas con mayor disponibilidad de memoria (V_1, V_2, H_1 y H_2), 97.74% al 98.58%, mientras que en las plataformas con menor memoria (W_1 y W_2) la repetitividad en Mosca de la Fruta fue de 95.51% y 95.97%, respectivamente. Para Pulga de Agua la repetitividad observada en las plataformas con mayor memoria fue del 89.32% al 93.85%, mientras que la repetitividad en las plataformas W_1 y W_2 fue 77.55% y 78.36%. Esto puede ser observado en los conjuntos de *contigs* intersectados de las Figuras 3 y 4, y en la Tabla 4.

Por otro lado, las plataformas con menor memoria presentaron mayor variabilidad (ver Tabla 4). Para Mosca de la Fruta la mayor variabilidad observada en las plataformas de baja memoria fue de 4.49% (en W_1); en el caso de las plataformas HPC la variabilidad máxima fue de 2.26% (mismo porcentaje en H_1 y H_2). Para Pulga de Agua las plataformas de menor memoria presentaron una variabilidad máxima de 22.45% (en W_2), mientras que para HPC fue del 10.68% (en H_2). Dadas estas variabilidades se observa que las plataformas menores presentan aproximadamente el doble de variabilidad en comparación de las plataformas de mayor memoria, 1.98 y 2.10 veces mayor en Mosca de la Fruta y Pulga de Agua respectivamente, de acuerdo con el cálculo de ganancia por variabilidad entre plataformas de la sección 2.4.1.

Las plataformas con menos memoria produjeron mayor cantidad de combinaciones de *contigs*. Sin embargo, se necesita determinar si estas secuencias, producto de un algoritmo computacional, tienen correspondencia a un transcrito real.

Como se mencionó con anterioridad, la concordancia entre ensamblajes y los transcriptomas originales se determinó por medio de mapeos a la referencia codificante más próxima de los organismos. De acuerdo con el análisis de resultados realizado, los mapeos de los conjuntos de *contigs* no intersectados fueron mayores en los conjuntos provenientes de plataformas con poca memoria (Figuras 5 y 6).

También se observa en la Tabla 5, que para la Mosca de la Fruta los mapeos máximos de los conjuntos no intersectados fueron de 99.16% para plataformas con poca memoria (en W_1), mientras que en HPC fue de 98.48% (en H_1). Para la Pulga de Agua los mapeos máximos de los conjuntos no intersectados fue 76.64% (en W_1), mientras que en HPC fue 71.52%(en H_2). Si bien los porcentajes de mapeos son similares para ambos organismos, los *contigs* no intersectados mapeados en las plataformas de menor memoria son aproximadamente el doble en comparación con los *contigs* mapeados en las plataformas HPC.

Para Mosca de la fruta la cantidad de *contigs* mapeados fue 2.03 veces mayor en la estación de trabajo comparando con los mapeos máximos para HPC; en Pulga de Agua mapearon 2.39 veces más *contigs* en la estación de trabajo comparando con los mapeos máximos en HPC.

Se puede observar también que los porcentajes de mapeos exclusivos provenientes de los conjuntos no intersectados $\bar{I}m^{+}_{(p,m)}$son similares para ambos organismos y que estos representan la proporción de información validada generada de manera exclusiva por una plataforma dada. Dichos *contigs* variaron desde el 29.15% al 30.63% en Mosca de la Fruta y del 29.35% al 33.76% en Pulga de Agua. Sin embargo, las ganancias en información nueva, que se basan en los *contigs* mapeados exclusivos $\bar{I}m^{+}_{(p,m)}$, indican que para Mosca de la Fruta la estación de trabajo obtuvo 1.73 veces más *contigs* mapeados exclusivos que las plataformas HPC; asimismo, en Pulga de Agua se obtuvieron en la estación de trabajo 2.46 veces más *contigs* mapeados exclusivos en comparación con las plataformas HPC.

Cabe mencionar que una referencia constituye la mejor aproximación disponible, en determinada fecha, al transcriptoma de estudio. En este contexto, se puede decir que gran porcentaje de los *contigs* obtenidos en los ensamblajes de ambos organismos corresponden a transcritos reales.

Algunos *contigs* no lograron ser mapeados a sus referencias. Sin embargo, no se puede asegurar que dichas secuencias no tengan correspondencia a algún transcrito, ya que las referencias son actualizadas según se profundice en el conocimiento de cierto organismo. Se tiene mayor conocimiento de las secuencias transcriptómicas en Mosca de la Fruta que de los transcritos de Pulga de Agua, ya que *Drosophila melanogaster* es uno de los organismos modelo más estudiados por la comunidad científica. El hecho de que los mapeos en Pulga de Agua sean menores no significa que las secuencias no mapeadas sean incorrectas, simplemente las secuencias de dichos *contigs* pueden ser parte de transcritos que aún no son descubiertos por falta de conocimiento sobre la biología molecular del organismo.

Asimismo, se puede mencionar que las ganancias aquí indicadas se presentaron al trabajar en plataformas computacionales con disponibilidad baja de memoria, pero con ~3 veces más que el mínimo teórico reportado[15], bajo las condiciones de viabilidad de plataforma para el procesamiento de conjuntos de lecturas de 7-8 millones de lecturas pareadas y con la unión sin repeticiones de *contigs* provenientes de 5 ensamblajes.

Dados estos resultados, se evidenció que la variación de un ensamblaje está dada en función de la disponibilidad de memoria del equipo de cómputo; a mayor disponibilidad de memoria menos variación en ensamblaje y a menor disponibilidad de memoria mayor variación en ensamblaje.

Una de las principales ventajas del RNA-Seq (secuenciación NGS de Transcriptoma) es el poder descubrir nuevos transcritos [6]. Tomando esta característica en cuenta y con base en los resultados de este estudio, se sugiere el emplear una estrategia repetitiva de ensamblaje *de novo* de transcriptoma para el descubrimiento de una mayor cantidad de transcritos.

Dicha estrategia consiste en la obtención de varios ensamblajes bajo condiciones iniciales iguales en plataformas computacionales viables para proceso, pero con baja disponibilidad de memoria; posteriormente, realizar la unión no repetitiva de *contigs* de múltiples ensamblajes, ya que se logra obtener conjuntos más grandes de *contigs* de alta calidad, como fue realizado en los *contigs* obtenidos en las plataformas de menor memoria W_1 y W_2 (ver Figuras 5 y 6).

El efecto de memoria en otras métricas distintas a la cantidad de *contigs* o su contenido no fueron analizadas en este estudio. Diversas métricas estadísticas buscan dar un indicativo de la calidad de ensamblaje con respecto al transcritptoma original, pero éstas métricas no muestran indicios cuantitativos del desempeño del equipo de cómputo o su influencia en el conjunto de *contigs*.

Del mismo modo, se hubiese esperado que la repetitividad de las plataformas V_1 y H_1 fuese muy similar dado que tienen la misma cantidad de memoria (128 GB); no obstante, se encontró mayor variación en la plataforma H_1. Esto pudiese sugerir que, aparte de la memoria RAM disponible para los procesos de ensamblaje, los efectos computacionales en ensamblaje pueden ser influidos en menor proporción por otros recursos, como la memoria cache del procesador (que es menor en el nivel L3 para los procesadores de las plataformas H_1 y H_2). Éste y otros aspectos, como el número de núcleos de procesamiento, necesitan ser estudiados para ampliar el conocimiento de los efectos computacionales en la tarea de ensamblaje.

5. Conclusiones

El ensamblaje *de novo* de transcriptoma es una etapa clave en estudios exploratorios del contenido de ARN. Es necesario tomar en cuenta los efectos que la plataforma computacional tiene sobre este proceso. Así mismo, el aprovechamiento de las plataformas disponibles para trabajos de investigación prospectiva debe ser potencializado en la etapa de ensamblaje. Un ensamblaje varía en función de la disponibilidad de memoria del equipo de cómputo; menor disponibilidad de memoria origina mayor variación

o combinaciones en múltiples ensamblajes. Aún más, los *contigs* extra originados por dicha variación han mostrado tener correspondencia con transcriptomas de referencia. Tomando ventaja de la viabilidad de datos, disponibilidad de plataforma y recursos computacionales y su influencia en variabilidad del ensamblaje en función de la memoria, se sugiere descubrir mayor cantidad de *contigs* de calidad al realizar n repeticiones de ensamblajes bajo las mismas condiciones iniciales.

Agradecimientos

Los autores agradecen al Laboratorio Nacional de Supercómputo del Sureste de México (LNS), perteneciente al padrón de laboratorios nacionales CONACYT, por los recursos computacionales, el apoyo y la asistencia técnica brindados, a través del proyecto No. 2016-028. También agradecen a Jorge Mario Rodríguez Meza, Roberto González Castellanos, Luis Carlos Moreno Galván, y María Isabel Castro Hernández del CIBNOR, por su apoyo técnico en redes de comunicaciones. La autora P.C.L. también agradece a CONACyT por su beca 362054. Asimismo, a *Penguin Computing On-Demand* por su soporte técnico. Finalmente, también agradece el soporte técnico proporcionado por el grupo de ayuda a usuarios del software ensamblador Trinity.

Referencias

1. **Krebs, J. E., Lewin, B., & Kilpatrick, S. T. (2014).** *Lewin's genes* XI.
2. **Romero-Vivas, E., Von Borstel-Luna, F. D., Gutiérrez-Jagüey, J., & et al. (2012).** Análisis de información genómica: Investigación Bioinformática (CIBNOR). *Ciencia, Tecnología e Innovación Para el Desarrollo de México,* pp. 119.
3. **Mardis, E. R. (2008).** Next-generation DNA sequencing methods. *Annual Review of Genomics and Human Genetics,* Vol. 9, pp. 387–402. DOI: 10.1146/annurev.genom.9.081307.164359.
4. **Baker, M. (2012).** De novo genome assembly: what every biologist should know. *Nature Methods*, Vol. 9, No. 4, pp. 333–337.
5. **Lodish, H., Berk, A., Kaiser, C. A., Scott, M. P., Zipursky, L., & Darnell, J. (2016).** *Molecular Cell Biology.*
6. **Korf, I. (2013).** Genomics: the state of the art in RNA-Seq analysis. *Nature Methods*, Vol. 10, pp. 1165–1166. DOI: 10.1038/nmeth.2735.
7. **Miller, J. R., Koren, S., & Sutton, G. (2010).** Assembly algorithms for next-generation sequencing data. *Genomics*, Vol. 95, pp. 315–327. DOI: 10.1016/j.ygeno.2010.03.001.
8. **Compeau, P. E. C., Pevzner, P., & Tesler, G. (2011).** How to apply de Bruijn graphs to genome assembly. *Nature Biotechnology*, Vol. 29, pp. 987–991. DOI: 10.1038/nbt.2023.
9. **Earl, D., Bradnam, K., St John, J., & et al. (2011).** Assemblathon 1: a competitive assessment of de novo short read assembly methods. *Genome Research*, Vol. 21, pp. 2224–2241.
10. **Bradnam, K. R., Fass, J. N., Alexandrov, A., & et al. (2012).** Assemblathon 2: evaluating de novo methods of genome assembly in three vertebrate species. *Gigascience*, Vol. 2, pp. 10. DOI: 10.1186/2047-217X-2-10.
11. **Salzberg, S. L., Phillippy, A. M., Zimin, A., Puiu, D., Magoc, T., Koren, S., Treangen, T. J., Schatz, M. C., Delcher, A. L., Roberts, M., Marçais, G., Pop, M., & Yorke, J. (2012).** GAGE: A critical evaluation of genome assemblies and assembly algorithms. *Genome Research*, Vol. 22, pp. 557–567. DOI: 10.1101/gr.131383.111.
12. **Lin, Y., Li, J., Shen, H., Zhang, L., Papasian, C. J., & Deng, H. W. (2011).** Comparative studies of de novo assembly tools for next-generation sequencing technologies. *Bioinformatics*, Vol. 27, No. 15, pp. 2031–2037. DOI: 10.1093/ bioinformatics/btr319.
13. **Zhao, Q. Y., Wang, Y., Kong, Y. M., Luo, D., Li, X., & Hao, P. (2011).** Optimizing de novo transcriptome assembly from short-read RNA-Seq data: a comparative study. *BMC Bioinformatics*, Vol. 12, No. 14. DOI: 10.1186/1471-2105-12-S14-S2.
14. **Henschel, R., Nista, P. M., Lieber, M., & et al. (2012).** Trinity RNA-Seq assembler performance optimization. *Proceedings of the 1st Conference on Extreme Science and Engineering Discovery Environment on Bridging from the eXtreme to the campus and beyond. (XSEDE'12)*, Vol. 8. DOI: 10.1145/2335755.2335842.
15. **Haas, B. J., Papanicolaou, A., Yassour, M., Grabherr, M., Blood, P. D., Bowden, J., Couger, M. B., Eccles, D., Li, B., Lieber, M., MacManes, M. D., Ott, M., Orvis, J., Pochet, N., Strozzi, F., Weeks, N., Westerman, R., William, T., Dewey, C. N., Henschel, R. D., LeDuc, R., Friedman, N., &**

Regev, A. (2013). De novo transcript sequence reconstruction from RNA-seq using the Trinity platform for reference generation and analysis. *Nature Protocols*, Vol. 8, pp. 1494–1512. DOI: 10.1038/nprot.2013.084.

16. **QUIAGEN Company (2014).** *Manual for CLC Genomics Workbench 7.0.*
17. **Grabherr, M. G., Haas, B. J., Yassour, M., & et al (2011).** Full-length transcriptome assembly from RNA-Seq data without a reference genome. *Nature Biotechnology*, Vol. 29, pp. 644–652. DOI: 10.1038/nbt.1883.
18. **BUAP (2017).** Laboratorio Nacional de Supercómputo del Sureste de México.
19. **POD (2017).** *Penguin Computing on Demand*. URL: https://pod.penguincomputing.com.
20. **Yoo, A. B., Jette, M. A., & Grondona, M. (2003).** SLURM: Simple Linux Utility for Resource Management. *Job Scheduling Strategies for Parallel Processing*, pp. 44–60. DOI: 10.1007/10968987_3.
21. **Staples G. (2006).** TORQUE Resource Manager. *Proceedings of the ACM/IEEE Conference on Supercomputing*.
22. **The MathWorks Inc. (2013).** *MATLAB and Bioinformatics Toolbox Release 2013a*.
23. **Leinonen, R., Sugawara, H., & Shumway, M., (2011).** The Sequence Read Archive. *Nucleic Acids Research*. Vol. 39, No. 1, pp. D19–D21. DOI: 10.1093/nar/gkq1019.
24. **Daines, B., Wang, H., Wang, L., & et al. (2011).** The Drosophila melanogaster transcriptome by paired-end RNA sequencing. *Genome Research*, Vol. 21, pp. 315–324. DOI: 10.1101/gr.107854.110.
25. **Leinonen, R., Akhtar, R., Birney, E., & et al. (2011).** The European Nucleotide Archive. *Nucleic Acids Research*, Vol. 39. DOI: 10.1093/nar/gkq967.
26. **Rozenberg, A., Parida, M., Leese, F., & et al. (2015).** Transcriptional profiling of predator-induced phenotypic plasticity in Daphnia pulex. *Frontiers in Zoology*, Vol. 12, No. 18. DOI: 10.1186/s12983-015-0109-x.
27. **Andrews, S. (2015).** *FastQC A Quality Control tool for High Throughput Sequence Data*.
28. **Bolger, A., Lohse, M., & Usadel, B. (2014).** Trimmomatic: a flexible trimmer for Illumina sequence data. *Bioinformatics*, Vol. 130, No. 15, pp. 2114-2120. DOI: 10.1093/bioinformatics/btu170.
29. **Flicek, P., Amode, M. R., Barrell, D., & et al. (2014).** Ensembl 2014. *Nucleic Acids Research*, Vol. 42, pp. 749–755. DOI: 10.1093/nar/gkt1196.
30. **Vezzi, F., Narzisi, G., & Mishra, B. (2012).** Reevaluating assembly evaluations with feature response curves: GAGE and assemblathons. *PLoS One*. DOI: 10.1371/journal.pone.0052210.
31. **Camacho, C., Coulouris, G., Avagyan, V., et al. (2009).** BLAST+: architecture and applications. *BMC Bioinformatics*, Vol. 10, No. 1. DOI: 10.1186/1471-2105-10-421.
32. **Wheeler, D. L. (2003).** Database resources of the National Center for Biotechnology. *Nucleic Acids Research*, Vol. 31, pp. 28–33. DOI: 10.1093/nar/gkl1031.

Article received on 23/01/2018; accepted on 02/04/2018.
Corresponding author is Patricia Carvajal López.