Learning Running-time Prediction Models for Gene-Expression Analysis Workflows

D. A. Monge, M. Holec, F. Železný and C. G. Garino

***Abstract*— One of the central issues for the efficient management of Scientific workflow applications is the prediction of tasks performance. This paper proposes a novel approach for constructing performance models for tasks in data-intensive scientific workflows in an autonomous way. Ensemble Machine Learning techniques are used to produce robust combined models with high predictive accuracy. Information derived from workflow systems and the characteristics and *provenance* of the data are exploited to guarantee the accuracy of the models. A gene-expression analysis workflow application was used as case study over homogeneous and heterogeneous computing environments. Experimental results evidence noticeable improvements while using ensemble models in comparison with single/standalone prediction models. Ensemble learning techniques made it possible to reduce the prediction error with respect to the strategies of a single-model with values ranging from 14.47% to 28.36% for the homogeneous case, and from 8.34% to 17.18% for the heterogeneous case.**

***Keywords*— Performance Prediction, Ensemble Learning, Workflows, Bioinformatics, Distributed Computing.**

I. Introducción

LA TECNOLOGÍA de flujos de trabajo está destinada a facilitar el desarrollo de aplicaciones a través de la combinación de componentes de software reutilizables. Esta tecnología permite el desarrollo de aplicaciones a gran escala por personas con poca o incluso nula experiencia en lenguajes de programación. Esta es una de las razones por la cual dicha tecnología ha sido ampliamente adoptada en muchas áreas científicas [16].

Dada la envergadura de las aplicaciones, la ejecución es coordinada por sistemas de gestión de flujos de trabajo (o WMS por sus siglas en inglés) que abstraen al usuario de los detalles de la infraestructura de computación subyacente. Este aspecto es muy importante para la ejecución de aplicaciones a gran escala debido a que los usuarios pueden tomar ventaja de una enorme potencia de cálculo (e.g. clusters, grids o clouds) sin necesidad de lidiar con las particularidades de la infraestructura subyacente.

D. A. Monge, ITIC Research Institute & Facultad de Ciencias Exactas y Naturales, National University of Cuyo (UNCuyo), Argentina dmonge@uncu.edu.ar

M. Holec, IDA Research Group, Czech Technical University, Czech Republic holecmat@fel.cvut.cz

F. Železný, IDA Research Group, Czech Technical University, Czech Republic zelezny@fel.cvut.cz

C. G. Garino, ITIC Research Institute & Facultad de Ingeniería, National University of Cuyo (UNCuyo), Argentina cgarcia@itu.uncu.edu.ar

Sin embargo, la gestión eficiente de las aplicaciones por parte de los WMSs requiere de estimaciones precisas de los tiempos de ejecución de las tareas. Estas estimaciones son provistas por modelos de desempeño que realizan predicciones acerca de los tiempos de ejecución de las tareas. Dichas predicciones permiten entre otras cosas la correcta planificación de tareas [3], el cumplimiento de requerimientos de calidad de servicio [4], o el autoescalado en infraestructuras cloud entre otros [9, 12].

Aunque los métodos de predicción utilizados por los WMSs proporcionan estimaciones precisas, requieren la supervisión de un experto para la construcción y puesta a punto de los modelos. Para hacer frente a este tipo de limitación, muchos autores se han focalizado en la utilización de estrategias de aprendizaje automático para generar los modelos de predicción de manera (semi-)automática. Siguiendo esta línea de pensamiento, este trabajo propone un nuevo método para la generación autónoma de modelos de desempeño utilizando técnicas de aprendizaje conjunto (o Ensemble Learning en inglés) las cuales permiten la construcción de modelos combinados de gran robustez. El objetivo final de nuestro enfoque es la minimización del esfuerzo humano requerido para construir los modelos pero sin perder calidad en las predicciones realizadas. Para lograr tal objetivo este trabajo utiliza la información de desempeño disponible en los registros de ejecución de los WMSs así como la información de proveniencia de los datos en la aplicación. Este trabajo extiende un estudio que hemos llevado a cabo recientemente [13] mediante el análisis detallado de las predicciones realizadas por los modelos combinados, así también como el estudio de la distribución de los errores de predicción individuales. A su vez, unos resultados más completos pueden encontrarse en [14].

El resto de este trabajo se organiza de la siguiente manera. En la sección II se proporciona una revisión de las estrategias de predicción de rendimiento basadas en métodos de aprendizaje automático. La sección III presenta el enfoque propuesto para el aprendizaje de modelos de predicción. La sección IV describe una serie de conjunto de flujos de trabajo de bioinformática así como la metodología utilizada para la validación de nuestra propuesta. La sección V presenta y discute los resultados obtenidos. Finalmente, conclusiones y trabajos futuros se discuten en la sección VI.

II. Antecedentes

La predicción del rendimiento de las aplicaciones ha sido estudiada desde la génesis de la computación paralela y distribuida [1]. Muchas de estas estrategias utilizan datos históricos para realizar predicciones evitando la necesidad de

tener que construir los modelos manualmente. Para ello, técnicas estadísticas y de aprendizaje automático permiten la derivación de modelos. Este enfoque supone una ventaja importante para las aplicaciones de flujos de trabajo que se ejecutan en entornos grid o cloud, dado que los modelos pueden ser refinados sin que el usuario tenga que supervisar el proceso o realizar tareas tediosas como por ejemplo hacer perfiles (benchmarks) de los recursos o de la ejecución de las aplicaciones, análisis de código fuente, etc.

Algunas de la estrategias para abordar el problema de predicción de rendimiento es mediante la estrategia k-vecinos cercanos [8, 11]. Las predicciones se realizan buscando ejemplos de ejecución previos con una configuración similar y utilizando dichos valores como predicción. Otros autores utilizan árboles de regresión para predecir el rendimiento de las aplicaciones [15]. Asimismo se han utilizado Redes Neuronales en otros contextos diferentes al de estimaciones del tiempo de ejecución de las tareas, como la realización de predicciones del precio de recursos cloud [18].

Existen dos aspectos fundamentales que limitan el desempeño de los modelos y que se describen a continuación. En primer lugar las técnicas discutidas se han utilizado teniendo en mente aplicaciones de cálculo intensivo descartando aspectos importantes como el tamaño o la estructura de los datos procesados. Este último aspecto es fundamental para lograr predicciones precisas en el contexto de flujos de trabajo científico dado que actualmente el volumen de datos procesados está aumentando de manera sin precedentes [6, 10].

En segundo lugar estas estrategias se basan en el uso de modelos independientes para realizar las predicciones. En el área de aprendizaje automático se sabe que la combinación de múltiples modelos generalmente permite alcanzar un mayor rendimiento que el uso de modelos independientes [17]. La siguiente lista resume algunas de las principales limitaciones de las estrategias de predicción basadas en aprendizaje automático examinadas:

- No aprovechan información sobre la procedencia de los datos y otros atributos de los mismos. Los atributos de los datos son una fuente central de información para lograr predicciones de desempeño de tareas de calidad.
- Las técnicas revisadas en esta sección se basan en un único modelo para predecir los tiempos de ejecución de las tareas.

Nuestra contribución este trabajo propone una nueva estrategia para reducir al mínimo la intervención de un experto humano en el modelado del desempeño de tareas dentro de flujos de trabajo científicos. La estrategia propuesta utiliza métodos de aprendizaje automático de conjunto (Ensemble Learning) para la generación de modelos de forma autónoma. Para ello se incorporan varias fuentes de información proporcionada por el WMS, tales como parámetros de tareas, información de hardware, atributos de los datos y la información de procedencia para maximizar la precisión de los modelos.

III. Generación Autónoma de Modelos.

Esta sección describe una estrategia novedosa para la generación autónoma de los modelos de rendimiento (AGPM por sus siglas en inglés) para la predicción de las tareas de flujos de trabajo en tiempo de ejecución. Como premisa para el desarrollo de esta propuesta, AGPM se basa únicamente en la información que puede ser accesible desde el sistema de administración de flujos de trabajo. En este sentido, sólo se requiere que el usuario defina los meta-datos que pueden ser importantes para el modelado del rendimiento de las tareas. De esta manera, el proceso de modelado se centra en los parámetros y datos que afectan al rendimiento (conocimiento empírico del usuario) y no en la lógica implementada por las tareas. Esta es una de las principales ventajas de AGPM dado que considera las tareas como cajas negras, permitiendo el modelado de desempeño de tareas cuyo código no se encuentra disponible o es inaccesible.

Para responder al requisito de *simplicidad de utilización*, AGPM se vale de métodos de aprendizaje automático los cuales permiten construir los modelos de manera autónoma. Para ello utiliza la información de los parámetros de las tareas, atributos de los datos, información de las dependencias entre tareas y métricas de rendimiento de los recursos. Los métodos de aprendizaje automático permiten la construcción de los modelos y su reajuste a medida que nuevos datos se encuentran disponibles. De esta manera, el esfuerzo humano necesario para mantener los modelos es reducido considerablemente.

A. Proceso Adaptativo

AGPM lleva a cabo un proceso de aprendizaje adaptativo que consta de 4 fases (ver Fig. 1): (i) la ejecución de las tareas, (ii) la recopilación de datos de rendimiento, (iii) el aprendizaje de los modelos, y (iv) la predicción del tiempo de ejecución de las tareas. En los párrafos siguientes se discuten brevemente cada una de estas etapas.

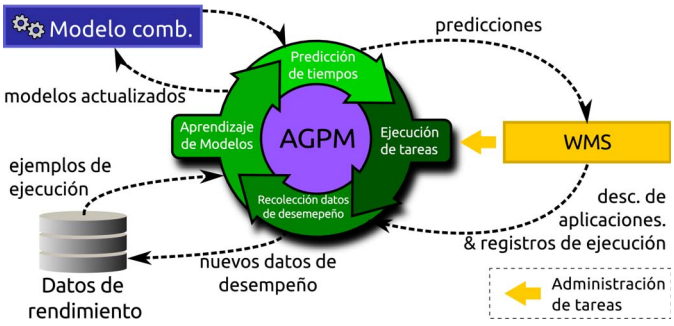


Figura 1. Proceso de aprendizaje llevado a cabo por AGPM.

a) Etapa 1: Ejecución de las tareas

Esta etapa comprende la ejecución y monitoreo de las tareas. En esta etapa se da también la generación de los registros de ejecución por parte del WMS. Cabe destacar que esta etapa es llevada a cabo en su totalidad por el WMS.

b) Etapa 2: Recopilación de datos de desempeño

Consiste en la recolección de la información necesaria para el aprendizaje y posterior refinamiento de los modelos de

rendimiento. Los registros de ejecución se utilizan para extraer información valiosa de desempeño de las tareas como los parámetros y los datos utilizados, información sobre la procedencia y las características de los recursos que ejecutaron dichas tareas. AGPM recopila toda la información que puede ser obtenida del WMS. Los datos recogidos se almacenan en bases de datos separadas para cada tipo de tarea ejecutable.

c) Etapa 3: Aprendizaje de modelos

En este punto del proceso, las bases de datos contienen información actualizada de las últimas ejecuciones de tareas. Esta información se utiliza para aprender nuevos modelos mediante un procedimiento de dos pasos que consiste en (i) el pre-procesamiento de los datos, y (ii) el aprendizaje conjunto de modelos. AGPM preprocesa las bases de datos con el fin de preparar los datos para la estrategia de aprendizaje conjunto. Como segundo paso, se aprenden los modelos de rendimiento a partir de los datos recopilados para poder realizar las predicciones futuras. Las secciones III-B y III-C proporcionan una visión más profunda sobre el proceso descrito, el cual es la principal contribución de este trabajo.

d) Etapa 4: Predicción del tiempo de ejecución de tareas

Consiste en la generación de estimaciones de tiempo para ejecutar las tareas utilizando los modelos generados en la etapa anterior. Las estimaciones de los tiempos de ejecución se obtienen considerando las entradas de tareas (es decir, parámetros y datos) y las características de los recursos que eventualmente ejecutarán dichas tareas.

Esta secuencia de etapas se repite continuamente durante la ejecución de las aplicaciones. Cada uno de estos ciclos permite la mejora de la exactitud de predicción de los modelos. Esta estrategia permite la adaptación de los modelos a nuevos ejemplos de ejecución aún no conocidos. El aspecto importante a destacar es que este proceso de aprendizaje adaptativo mejora la precisión de los modelos de predicción, sin necesidad de intervención humana a excepción de la especificación de cuales datos son importantes para el modelado del desempeño. El aprendizaje conjunto juega un papel central en este proceso porque provee modelos robustos de manera autónoma.

B. Representación de los Datos

Los datos de rendimiento se almacenan por separado para cada tipo de tarea. El conjunto de datos de rendimiento para una tarea se puede definir formalmente como un conjunto $D = \{x^{(i)}, y^{(i)}\}$, donde $x^{(i)}$ representa un vector columna con los atributos para el i-ésimo ejemplo de ejecución (de entre m), e $y^{(i)}$ es el tiempo de ejecución medido para dicha ejecución, también conocido como objetivo.

Cada vector de atributos $x = [x_1, x_2, \dots, x_n]$ comprende tres tipos de elementos: (i) atributos de las *tareas*, que representan las entradas de la tarea, por ejemplo, valores de los parámetros, tamaño de los datos, etc.; (ii) los atributos de *procedencia*, describen los procesos que tuvieron lugar previamente y que generaron o modificaron los datos de entrada; y (iii) los atributos de los *recursos* utilizados en la ejecución de las tareas.

Atributos de las tareas

Este tipo de atributos describen las entradas de la tarea. Esta información incluye los valores que toman los parámetros de entrada y los atributos de los datos, tales como el tamaño, número de líneas, número de columnas, etc.

Atributos de procedencia

Este tipo de atributos capturan información del origen de los datos y las transformaciones producidas por otras tareas durante la ejecución de la aplicación. Dicha información puede ser fácilmente extraída a partir de la descripción de la aplicación. Como se dijo antes, hasta donde tenemos noción, no existen otras estrategias que utilicen dicha información para producir predicciones del tiempo de ejecución.

Atributos de Recursos

Este tipo de atributos describen los recursos computacionales utilizados en la ejecución de las tareas. Estos atributos se pueden obtener de los sistemas de administración de flujos de trabajo. Los atributos utilizadas para modelar el tiempo de ejecución de las tareas son aquellas que miden el desempeño de los recursos. Dicha información es proporcionada principalmente por benchmarks de los recursos. En general, la mayor parte de los WMSs proporcionan dichos indicadores y los actualizan regularmente.

C. Aprendizaje de Modelos

Como mencionamos anteriormente, los métodos de aprendizaje automático son el núcleo de nuestro enfoque. En esta sección se describen brevemente algunas de las técnicas de aprendizaje automático tradicionalmente utilizadas para generar los modelos (independientes) de predicción. Esta sección también describe una estrategia de aprendizaje conjunto denominada Bootstrap Aggregating (Bagging) utilizada para validar nuestra hipótesis.

1) Modelos Independientes

Nuestra implementación de AGPM incluye algunas de las estrategias de aprendizaje automático más utilizadas para la predicción del rendimiento. Los siguientes párrafos describen los conceptos fundamentales detrás de este tipo de estrategias. Tenga en cuenta que las implementaciones de los algoritmos utilizados en este trabajo son proporcionados por la biblioteca Weka [17].

a) k-Vecinos Cercanos

En esta estrategia, los ejemplos de entrenamiento se almacenan literalmente. Una función de distancia se utiliza para determinar aquellos k ejemplos del conjunto de entrenamiento que están más cerca de la instancia para la cual se quiere realizar la predicción [19]. La predicción realizada por el método es el promedio de los valores objetivo (es decir, tiempo de ejecución) correspondiente a los k ejemplos más cercanos. En este estudio utilizamos la distancia euclídea. Dados dos ejemplos $x^{(1)}$ y $x^{(2)}$ con componentes $x_1^{(1)}, x_2^{(1)}, \dots, x_n^{(1)}$ y $x_1^{(2)}, x_2^{(2)}, \dots, x_n^{(2)}$ respectivamente, la distancia euclídea entre ellos es $\sqrt{\sum_{i=1}^{n}(x_i^{(1)} - x_i^{(2)})^2}$.

b) Redes Neuronales

Este tipo de redes pertenecen a una familia de modelos que emulan el funcionamiento de las redes neuronales biológicas [19]. Los modelos comprenden un conjunto de

neuronas dispuestas en múltiples capas. Las neuronas en una capa están conectadas a las neuronas en la capa siguiente a lo largo de la red que alcanza una neurona de salida la cual predice el valor objetivo y. Las redes utilizadas en este estudio comprenden una capa oculta con $n/2$ neuronas ocultas, donde n es el número de atributos en el vector de entrada. Los parámetros de la red neuronal son una matriz $\Theta^{(1)} \in \mathbb{R}^{(n+1)\times(n/2)}$ y una matríz $\Theta^{(2)} \in \mathbb{R}^{(n/2+1)\times 1}$. Dichas matrices modelan las interacciones entre las neuronas en diferentes capas.

Las activaciones (o valores de salida) de las neuronas ocultas se calculan como $= \sigma(\bar{x}) \cdot \Theta^{(1)})$, donde $\bar{x}$ es un vector de atributos x extendido con un primer componente $x_0 = 1$ (neurona de sesgo), y $\sigma(z) = 1/(1 + e^{-z})$ es la función sigmoidea. La activación de las unidades ocultas se propaga hacia adelante a la siguiente capa para producir el valor pronosticado del tiempo de ejecución $y = \bar{a} \cdot \Theta^{(2)}$, donde $\bar{a}$ es el vector de activación de una de la capa oculta ampliado con una neurona de sesgo $a_0 = 1$. Aprender el modelo consiste en el aprendizaje de los pesos en la red, es decir, los valores de matrices $\Theta^{(1)}$ y $\Theta^{(2)}$. Para tal fin se utiliza el algoritmo de Backpropagation [19], el cual no será posteriormente analizado por estar fuera del alcance de este trabajo.

c) Máquinas de Vectores de Regresión

Los modelos Regresión con Vectores de Soporte (SVR) [19] son una adaptación de las Máquinas de Vectores Soporte (SVM) para hacer frente a la predicción de las clases numéricas. Los modelos obtenidos se pueden expresar en términos de los vectores de soporte que mejor describen una superficie de la predicción. El modelo SVR tiene la forma $f(x) = \sum_{i\in SV} \alpha_i K(x^{(i)}, x) + b$, donde SV es el conjunto de vectores de soporte y $K(x^{(i)}, x)$ es una función del kernel que mapea un ejemplo en un espacio de atributos de dimensiones superiores. α_i y b son parámetros del modelo los cuales se determinan resolviendo el siguiente problema de optimización:

$$\min_{\alpha,\beta,\xi_i,\xi_i^*} \frac{1}{2}\alpha^T\alpha + C\sum_{i=1}^{l}(\xi_i + \xi_i^*)$$

sujeto a:

$$\begin{aligned} y - f(x_i) &\leq \varepsilon + \xi_i \\ f(x_i) - y &\leq \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* &> 0, \end{aligned}$$

donde C es el parámetro de complejidad del modelo el cual rige las penalizaciones de los errores de entrenamiento, ξ_i y ξ_i^* son variables de holgura que especifican límites superiores e inferiores de los errores de entrenamiento sujetos a un error de tolerancia ε. Para modelar funciones no lineales del tiempo de ejecución de las tareas se utiliza un kernel de función de base radial (RBF por sus siglas en inglés) $K(x^{(i)}, x) = exp(-\gamma\left\|x^{(i)} - x^{(j)}\right\|^2)$ con $\gamma = 0{,}01$. Los parámetros de valores de SVR fueron establecidos a $C = 1$ y $\varepsilon = 0{,}001$.

d) Árboles de Regresión M5P

El algoritmo M5 Prime (M5P) [19] permite la construcción de árboles de decisión cuyas hojas están asociadas a modelos de regresión. Los árboles M5P son una combinación de árboles de decisión y modelos lineales en las hojas de los árboles. El modelo se construye en tres etapas. En primer lugar, se construye un árbol de decisión minimizando la variabilidad de la variable objetivo (es decir, ejemplos con tiempo de ejecución similar son agrupados). Luego, el árbol se poda a partir de los nodos hoja. Cuando se poda un nodo interno, este se convierte en un nodo de hoja al cual se asocia un modelo de regresión lineal (i.e. un hiperplano en el espacio de atributos utilizado para realizar las predicciones). Finalmente, se aplica una función de suavizado para evitar discontinuidades entre los hiperplanos generados.

2) Preproceso de Atributos

Antes de la construcción de los modelos, los datos de rendimiento disponibles se normalizan para evitar el predominio de atributos con grandes órdenes de magnitud durante la construcción de los modelos. Para tal fin, cada atributo x_i se transforma en un nuevo atributo $\hat{x}_i$ de acuerdo con la ecuación $\hat{x}_i = \frac{x_i - \mu}{\sigma}$, donde μ y σ son la media y la desviación estándar de todos los valores para el atributo x_i. En este trabajo, el proceso de atributos se limita a la normalización de los datos. Sin embargo queda pendiente la exploración de otras estrategias de preproceso y aprendizaje de atributos que podrían mejorar la calidad de los modelos [2].

3) Modelos Combinados

Una de las principales ventajas de los métodos de aprendizaje conjunto es que en general permiten obtener predicciones de mejor calidad que los obtenidos por los modelos independientes y nunca presentan un desempeño mucho peor. Para la generación de los modelos que utilizamos la técnica Bootstrap Aggregating (Bagging) [19]. Esta técnica permite reducir la varianza de los modelos, i.e. el error esperado derivado de todos los posibles conjuntos de entrenamiento para el problema.

La técnica mencionada funciona de la siguiente manera. Para un determinado conjunto de datos de entrenamiento D se obtienen n nuevos (sub-)conjuntos de entrenamiento (D_i) cada uno de tamaño m'. Estos se obtienen tomando muestras azar y con reemplazo del conjunto D (es decir, algunos ejemplos se eliminan y algunos se repiten). Cada una de las n muestras se utilizan para generar n diferentes modelos base. Las salidas de los n modelos se combinan a través de un promedio. Este procedimiento genera un modelo combinado que, como se describió anteriormente, por lo general supera a los modelos individuales y nunca es considerablemente peor que un modelo independiente. Como modelos base para este estudio se utilizaron árboles de regresión M5P. El proceso completo se ilustra en la Fig. 2.

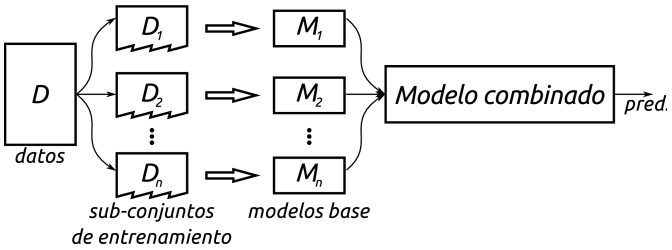


Figura 2. Proceso de Bagging. Las n muestras (D_i) se utilizan para contruir (M_i) modelos base, los cuales corresponden a árboles de regresión M5P.

IV. Experimentos

Para analizar el rendimiento de las distintas estrategias se evaluó la calidad predictiva de los modelos independientes generados así como de modelos aprendidos utilizando la estrategia de bagging. Las secciones siguientes describen una aplicación bioinformática utilizada como caso de estudio. Además se detalla la metodología utilizada para la validación de nuestra propuesta.

A. Análisis de Expresiones de Genes

A los efectos de este trabajo evaluamos nuestro enfoque en una aplicación que realiza un experimento de análisis de expresiones de genes (GEAE) de gran escala. El objetivo de dicho experimento es medir el desempeño de un algoritmo de clasificación novedoso (denominado GELF) [7] en su capacidad para clasificar con datos aún no conocidos.

El experimento comprende la ejecución de varios flujos de trabajo. Cada uno de ellos procesa uno de 20 conjuntos de datos usando un esquema de validación cruzada. Para obtener una estimación no sesgada del clasificador GELF el experimento lleva a cabo a 10 veces la validación cruzada dando lugar a 10 ejecuciones paralelas de cada sub-experimento. Como puede verse en la Fig. 3 cada sub-experimento implica 3 tipos de tareas descritas a continuación. La aplicación comprende dos tipos de tareas que realizan una selección de genes con el fin de reducir el número de atributos para el aprendizaje del clasificador. El primero utiliza una función recursiva de eliminación utilizando máquinas de vectores soporte [5], denominada SVM-RFE. La segunda devuelve un orden aleatorio de atributos y se denomina RandomR. La tercera tarea consiste en el aprendizaje y evaluación del desempeño de la tarea de clasificación, esta tarea se denomina GELF y será el objeto de análisis de este trabajo. GELF es un algoritmo de construcción de atributos basado en la mejora iterativa sobre la mejor solución obtenida por un enfoque alternativo utilizado como base [7].

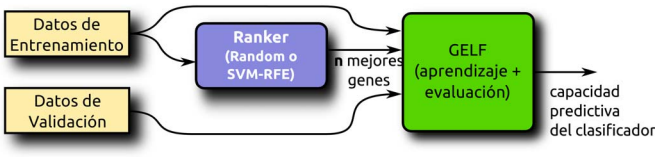


Figura 3. Descripción de un sub-experimento de los flujos de trabajo para el análisis de expresiones de genes. Cada sub-experimento opera sobre una de 10 porciones de un conjunto de datos particular. Considerando que hay 2 tipos de tareas para ranking de genes. Cada flujo de trabajo comprende 20 sub-experimentos.

B. Configuración de los Experimentos

Se midió el tiempo de ejecución de múltiples instancias de la tarea GELF utilizando para ello 20 conjuntos de datos de entrada diferentes y 26 recursos de computación descriptos en la Tabla I.

Tabla I. Resumen de la infraestructura de computación.

	Tipo de Recurso			
Características	Twister	Reloaded	Opteron	
Procesador	Intel Core2 Duo	Intel P4 HT	AMD Opteron 242	
Frecuencia	3.0 GHz	3.0 GHz	1.6 GHz	
Memoria	4 GB	1 GB	2 GB	
JavaMFlops	962.23	281.05	400.76	
KFlops	17.54E5	4.99E5	6.63E5	
MIPS	4983.80	1465.42	2057.00	
Cantidad	10	12	4	

Mediante la ejecución repetida de la aplicación, se generaron dos bases de datos de rendimiento comprendiendo 4000 instancias de ejecución cada una. La primera de ellas, denominada homogénea, contiene instancias de ejecución en donde solo se utilizaron recursos de tipo Twister (ver Tabla I). La segunda de las bases de datos, denominada heterogénea, contiene instancias de ejecución correspondientes a la utilización de todos los tipos de recursos disponibles. La Tabla II describe los atributos utilizados para cada instancia de ejecución de las tareas.

Tabla II. Caraterísticas utilizadas para el modelado del desempeño de las tareas de clasificación.

Atributo	Tipo	Descripción
dataset-id	proveniencia	set de datos {1,2,…,20}
ranker	proveniencia	ranking de genes {RandomR,SVM-RFE}
tr-size	tarea	tamaño en bytes del set de entrenamiento
tr-rows	tarea	# de filas del set de entrenamiento
tr-columns	tarea	# de columnas del set de entrenamiento
tt-size	tarea	tamaño en bytes del set de validación
tt-rows	tarea	# de filas del set de validación
tt-columns	tarea	# de columnas del set de validación
Java-mflops	recurso	resultado del benchmark SciMark2
kflops	recurso	resultado del benchmark Linpack
mips	recurso	resultado del benchmark Dhrystone
running-time	objetivo	tiempo de ejecución medido

Con dichas bases de datos se realizó la creación de los modelos utilizando todas las estrategias de aprendizaje analizadas previamente. El proceso de aprendizaje y evaluación de las estrategias se repitió 30 veces en cada tipo de entorno de computación (homogéneo/heterogéneo). En cada caso se utilizó un 70% de las (4000) instancias originales para el aprendizaje y el restante 30% para validación utilizando muestreos distintos de los datos.

V. Análisis de Resultados

En esta sección se presentan y discuten los resultados experimentales. Se analizó el desempeño de las estrategias de aprendizaje para la tarea GELF considerando infraestructuras homogéneas (considerando solo recursos de tipo Twister) y heterogéneas. Para medir el desempeño de cada modelo se utiliza el Error Absoluto Relativo (EAR), el cual se calcula

como $EAR = \frac{|p_1-a_1|+\cdots+|p_m-a_m|}{|a_i-\bar{a}|+\cdots+|a_m-\bar{a}|} \cdot 100\%$, donde p_i y a_i representan los valores predicho y real respectivamente, para el i-ésimo ejemplo. $\bar{a}$ representa el valor medio de los valores reales y m es el número de ejemplos de validación. Esta métrica mide la desviación de las predicciones con respecto a los valores reales.

A. Comparación de Estrategias

La tabla III muestra los valores medios, medianas, valores extremos y desviación estándar de los errores de predicción para cada tipo de estrategia de aprendizaje en los entornos homogéneo y heterogéneo. Los valores resaltados representan los errores mínimos para cada tipo de entorno. Puede observarse que para el caso homogéneo el error de predicción más bajo corresponde a la estrategia de aprendizaje conjunto. Dicha estrategia obtuvo un error de 28.85%. En comparación con los demás métodos, las mejoras del método de Bagging van desde un 14.47% a un 28.36%.

De manera similar, en el caso heterogéneo puede observarse que el error más pequeño de predicción corresponde a la estrategia de Bagging con un error de 22,36%. En comparación con los demás métodos, la estrategia de aprendizaje conjunto presenta reducciones de los errores de predicción que van desde un 8.34 % a un 17.18 %.

TABLA III. ERRORES PARA LOS ENTORNOS DE COMPUTACIÓN HOMOGÉNEO Y HETEROGÉNEO.

Entorno	Estrategia	media	mediana	min	max	desviación
Homog.	M5P	43.32	43.50	40.77	45.15	**1.07**
	ANN	57.21	54.18	46.59	82.49	10.38
	k-NN	44.55	44.44	42.49	46.87	1.12
	SVR	52.34	52.51	49.82	55.91	1.40
	Bagging	**28.85**	**28.72**	**27.13**	**31.36**	1.16
Heterog.	M5P	30.70	30.78	28.47	32.62	**0.98**
	ANN	38.19	35.97	30.74	65.44	6.64
	k-NN	31.24	31.49	28.91	33.22	1.14
	SVR	39.54	39.55	37.61	41.98	1.17
	Bagging	**22.36**	**22.06**	**20.57**	**31.48**	1.86

La Fig. 4 combina los errores para los dos tipos de escenarios (homogéneo y heterogéneo). En la figura pueden apreciarse que las mejoras evidenciadas por la estrategia de aprendizaje conjunto son de al menos un 12%. Es interesante también destacar que el mejor de los competidores es el algoritmo M5P, el cual es utilizado para construir los modelos base en Bagging. Con la simple combinación de modelos M5P se pueden lograr mejoras significativas en la calidad de las predicciones.

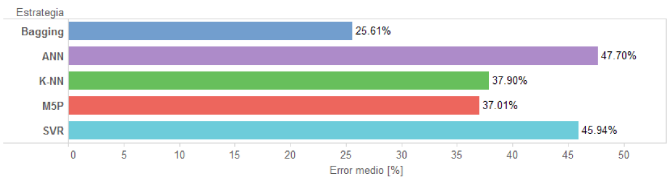


Figura 4. Errores obtenidos considerando los dos tipos de escenarios.

B. Predicciones del Modelo Combinado

La Fig. 5 muestra los valores reales de los tiempos de ejecución (eje horizontal) y los valores predichos por los modelos en el conjunto de validación (eje vertical) para los dos tipos de infraestructuras. Puede observarse que en general la mayoría de los puntos se encuentran distribuidos cerca de la recta de predicciones óptimas, lo cual significa que las predicciones tienen buena calidad, como se había observado previamente.

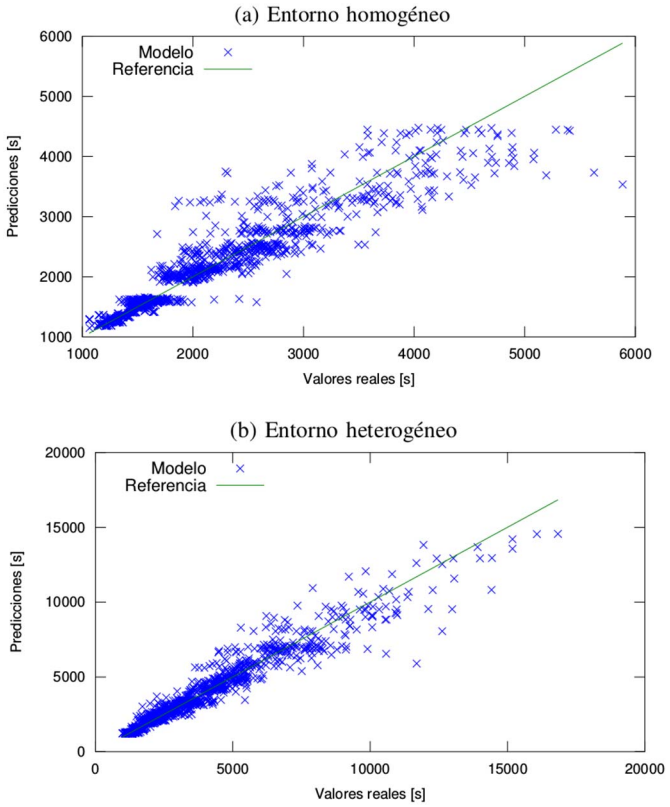


Figura 5. Predicciones realizadas para los entornos (a) homogéneo y (b) heterogéneo.

Otros aspectos interesantes a destacar son, en primer lugar, que para instancias de ejecución de mayor duración los modelos tienden a presentar una mayor dispersión, es decir, las predicciones tienden a alejarse de los valores reales. En segundo lugar, para dichas instancias de larga duración los modelos tienden a sub-estimar los valores reales.

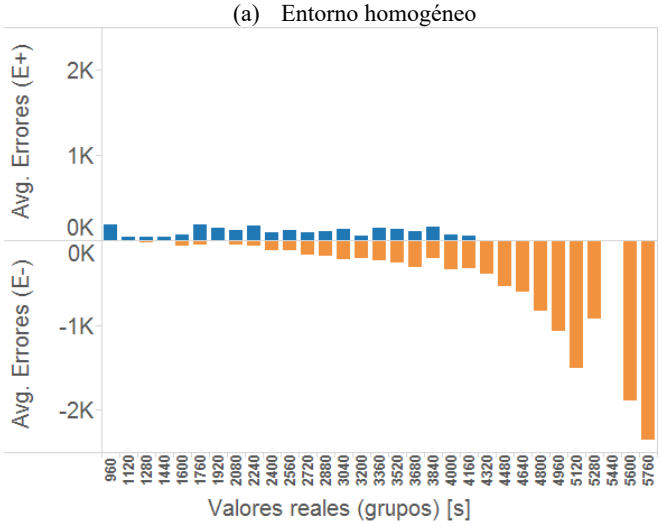
La Fig. 6 presenta los errores de sobre-estimación (E^+) y de sub-estimación (E^-) para los dos entornos, homogéneo y heterogéneo, agrupando las instancias de ejecución de acuerdo a las duraciones reales. El error en una predicción para el i-ésimo ejemplo es calculado como $e_i = p_i - a_i$, donde p_i y a_i representan los valores predicho y real respectivamente. Si el error es positivo ($e_i > 0$) el modelo incurrió en un error de sobre-estimación (E^+). De manera similar, si el error es negativo ($e_i < 0$) el modelo incurrió en un error de sub-estimación (E^-).

Para ambos tipos de entornos puede verse que los errores promedio de sobre-estimación son, en general, de menor magnitud que los errores de sub-estimación. Otro aspecto interesante a destacar es que para instancias de ejecución de mayor duración (más de 4160 s y 12075 s para las infraestructuras homogénea y heterogénea respectivamente) los modelos presentan solamente errores de sub-estimación de los tiempos de ejecución.

El comportamiento observado en ambos casos se deriva del hecho de que para duraciones mayores de las tareas existe una

menor cantidad de ejemplos con lo cual resulta más complicado a los modelos el caracterizar el rendimiento de manera apropiada. Esta sub-estimación de los tiempos de predicción da una pista para continuar con el mejoramiento de la calidad de las predicciones en trabajos futuros.

(a) Entorno homogéneo



(b) Entorno heterogéneo

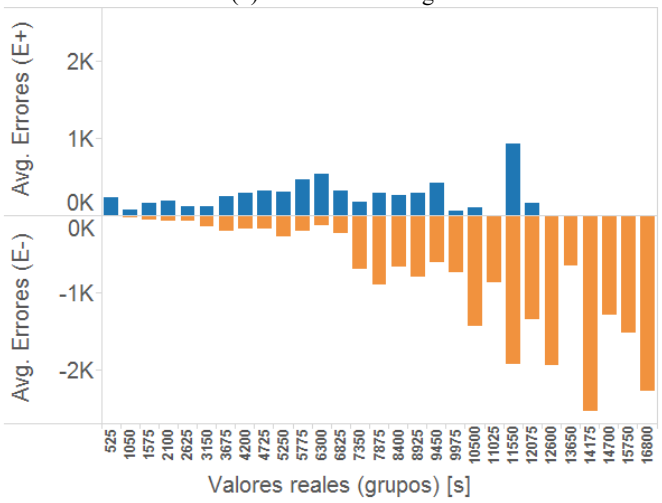


Figura 6. Errores para los entornos (a) homogéneo y (b) heterogéneo. Cada barrar representa el error promedio para el grupo de predicciones que correspondiente según los valores reales de tiempos de ejecución. Los errores de sobre-estimación (E^+) se presentan en azul y los de sub-estimación (E^-) en naranja).

VI. Conclusiones

En este trabajo se propone un esquema adaptativo para la construcción de modelos de rendimiento para aplicaciones de flujos de trabajo en infraestructuras grid o cloud. Este esquema utiliza métodos de aprendizaje automático para la construcción de modelos combinados utilizando información de datos de proveniencia y otras fuentes de información disponibles de los sistemas de administración de flujos de trabajo.

Los experimentos realizados fueron diseñados para evaluar el rendimiento de un conjunto de modelos en comparación con otras técnicas de aprendizaje automático de modelos independientes. Los experimentos se centraron en predecir el tiempo de ejecución de las tareas comprendidas en un una aplicación real de flujos de trabajo para el análisis de expresiones de genes. El rendimiento de las estrategias estudiadas se midió considerando entornos homogéneos y heterogéneos de computación.

El método propuesto presentó reducciones de error en el rango de 14.47% a 28.36% en el entorno homogéneo, y de 8.34% a 17.18% para el caso heterogéneo.

De un análisis un poco más detallado del desempeño de los modelos combinados puede observarse que los modelos tienden a subestimar los tiempos de ejecución para aquellos ejemplos de mayor duración. Este resultado motiva a continuar la investigación a fin de reducir los errores cuando la duración de las tareas se hace mayor.

Adicionalmente, como trabajo futuro tenemos la intención de evaluar otras estrategias de aprendizaje conjunto y analizarlas no solo en cuanto a la capacidad predictiva sino también en cuanto a los tiempos de aprendizaje y predicción. También resulta interesante explorar la aplicabilidad de estrategias de aprendizaje de representaciones.

Agradecimientos

El primer autor agradece al Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET) por la beca post-doctoral concedida. Esta investigación es apoyada por la Agencia Nacional de Promoción Científica y Tecnológica (ANPCyT) a través del proyecto Nº PICT-2012-2731. Se agradece al Ministerio de Argentino de Ciencia y Técnica (MINCyT) y al Ministerio Checo de Educación, Juventud y Deportes (MEYS) por los proyectos Nos. RC0904 y MEB111005. Se agradece también el apoyo financiero prestado por SeCTyP-UNCuyo a través del proyecto Nº M004.

Referencias

[1] R.J. Allan. Survey of HPC performance modelling and prediction tools. Technical Report DL-TR-2010-006, Science and Technology Facilities Council, Great Britain, July 2010.

[2] Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. Representation Learning: A Review and New Perspectives. Computing Reseach Repository-arXiv, abs/1206.5538:1–30, April 2014. http://arxiv.org/abs/1206.5538.

[3] Weiwei Chen and Ewa Deelman. Partitioning and scheduling workflows across multiple sites with storage constraints. In Parallel Processing and Applied Mathematics. Springer Berlin / Heidelberg, 2012.

[4] T.A.L. Genez, L.F. Bittencourt, and E. R M Madeira. Workflow scheduling for SaaS / PaaS cloud providers considering two SLA levels. In Network Operations and Management Symposium (NOMS), 2012 IEEE, pages 906–912, 2012.

[5] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene selection for cancer classification using support vector machines. Machine Learning, 46(1–3):389–422, March 2002.

[6] Tony Hey, Stewart Tansley, and Kristin Tolle, editors. The Fourth Paradigm: Data-Intensive Scientific Discovery. Microsoft Research, Redmond, Washington, October 2009.

[7] Matěj Holec, Jiří Klema, Filip Zelezný, and Jakub Tolar. Comparative evaluation of set-level techniques in predictive classification of gene expression samples. BMC Bioinformatics, 13, Suppl. 10(S15):1–15, 2012.

[8] M.A. Iverson, F. Ozguner, and L.C. Potter. Statistical prediction of task execution times through analytic benchmarking for scheduling in a heterogeneous environment. In Heterogeneous Computing Workshop. (HCW '99) Proceedings of the Eighth, volume 8, pages 99–111, San Juan, Puerto Rico, April 1999. IEEE Computer Society.

[9] Ming Mao and Marty Humphrey. Scaling and scheduling to maximize application performance within budget constraints in cloud workflows.

In Parallel & Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on, pages 67–78. IEEE, 2013.8
[10] Vivien Marx. Biology: The big challenges of big data. Nature, 498(7453):255–260, June 2013.
[11] David A. Monge, Jiří Bělohradský, Carlos García Garino, and Filip Železný. A Performance Prediction Module for Workflow Scheduling. In High-Performance Computing in Latin America (HPCLatAm 2011), 40 JAIIO, 4th Symposium on, 2011.
[12] David A. Monge and Carlos García Garino. Adaptive spot instances aware autoscaling for scientific workflows on the cloud. In G. Hernández et al., editor, High Performance Computing. Springer Berlin Heidelberg, 2014.
[13] David A. Monge, Matěj Holec, Filip Železný, and Carlos Garc´ıa Garino. Ensemble learning of run-time prediction models for data-intensive scientific workflows. In G. Hernández et al., editor, High Performance Computing. Springer Berlin Heidelberg, 2014.
[14] David A. Monge, Matěj Holec, Filip Železný, and Carlos Garc´ıa Garino. Ensemble Learning of Runtime Prediction Models for Gene-Expression Analysis Workflows. Cluster Computing, pages 1-13. Springer US, 2015.
[15] E. Ould-Ahmed-Vall, J. Woodlee, C. Yount, K.A. Doshi, and S. Abraham. Using model trees for computer architecture performance analysis of software applications. In Performance Analysis of Systems Software, 2007. ISPASS 2007. IEEE International Symposium on, pages 116–125. IEEE Computer Society, April 2007.
[16] I.J. Taylor, E. Deelman, D.B. Gannon, and M. Shields. Workflows for e-Science: Scientific Workflows for Grids. Springer London, 1st edition, December 2007.
[17] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P.,Witten, I.H.: The weka data mining software: an update. SIGKDD Explor. Newsl. 11, 10–18, 2009.
[18] R.M. Wallace, V. Turchenko, M. Sheikhalishahi, I Turchenko, V. Shults, J.L. Vazquez-Poletti, and L. Grandinetti. Applications of neural-based spot market prediction for cloud computing. In Intelligent Data Acquisition and Advanced Computing Systems (IDAACS), 2013 IEEE 7th International Conference on, volume 2, pages 710–716, September 2013.
[19] Ian H. Witten, Eibe Frank, and Mark A. Hall. Data Mining: Practical Machine Learning Tools an Techniques. Morgan Kaufman, 3rd edition, January 2011.

David A. Monge recibió el título de Ingeniero en Sistemas de Información de la Universidad Tecnológica Nacional, Mendoza, Argentina, en 2007 y recibió el título de Doctor en Ciencias de la Computación en la Universidad Nacional del Centro de la Provincia de Buenos Aires en 2013. Es miembro del Instituto para las Tecnologías de la Información y las Comunicaciones (ITIC), Universidad Nacional de Cuyo, Argentina. Sus intereses de investigación incluyen aprendizaje de máquinas, big data, grid y cloud computing.

Matěj Holec recibió el título de Doctor en el Departamento de Cibernética de la Facultad de Ingeniería Eléctrica de la Universidad Técnica Checa en Praga, República Checa, en 2015. Fue miembro del Grupo de Investigación de Análisis Inteligente de Datos en el Departamento de Ciencias de la Computación de la Universidad Técnica Checa en Praga, donde trabajó en el análisis de expresiones de genes utilizando aprendizaje de máquinas, en particular, la clasificación a nivel de conjunto de datos de expresión génica. Sus intereses de investigación incluyen el aprendizaje de máquinas y el descubrimiento de conocimiento en conjuntos de datos masivos.

Filip Železný es profesor de ciencias informáticas en la Universidad Técnica Checa en Praga. Anteriormente, fue investigador postdoctoral en la Universidad de Wisconsin-Madison y profesor visitante en SUNY Binghamton. Se interesa por el aprendizaje automático, la programación lógica inductiva y la bioinformática.

Carlos García Garino se graduó en Ingeniería en la Universidad de Buenos Aires, Argentina en 1978 y recibió título de doctor en la Universidad Politécnica de Cataluña, Barcelona, España, en 1993. En la actualidad es Profesor Titular de la Facultad de Ingeniería y Director del Instituto de Investigación ITIC, Universidad Nacional de Cuyo, Argentina. Sus intereses de investigación incluyen la Mecánica Computacional, Redes de Computadoras y Computación Distribuida.