Gestión Eficiente de Recursos Grid Basada en la Búsqueda Dispersa. Facilitando la Auto-adaptación de Aplicaciones

María Botón-Fernández[1], Miguel A. Vega-Rodríguez[2] y Francisco Prieto Castrillo[1]

Resumen—**Las infraestructuras de computación grid son sistemas compuestos por un conjunto de recursos heterogéneos y geográficamente dispersos. A pesar de las ventajas que proporciona este paradigma computacional, una serie de problemáticas relacionadas con la selección y disponibilidad de recursos se han convertido en un reto cada vez más estudiado por la comunidad científica.**

El objetivo de este trabajo es proporcionar una estrategia inteligente y auto-adaptativa para la selección de recursos grid durante la ejecución de aplicaciones. Esta capacidad de adaptación se consigue aplicando un método evolutivo conocido como Búsqueda Dispersa (*Scatter Search*) durante el proceso de selección. Para ello, los recursos serán seleccionados en base a una serie de restricciones y aplicando un conjunto de reglas de calidad y diversidad.

Finalmente, a partir de los resultados obtenidos en las pruebas efectuadas en una grid real, se puede deducir que el modelo propuesto realiza un uso efectivo de los recursos grid consiguiendo, además, una reducción del tiempo de ejecución de aplicaciones y un incremento en la tasa de tareas finalizadas con éxito. Como conclusión, el modelo mejora la productividad de las infraestructuras grid en beneficio de las aplicaciones.

Palabras clave— **Computación grid, Búsqueda dispersa, adaptabilidad, selección de recursos, optimización**

I. Introducción

LOS entornos de computación grid [1] [2] son sistemas distribuidos compuestos por muchos recursos de naturaleza heterogénea, con una disposición geográfica diferente y que trabajan conjuntamente para ejecutar tareas computacionalmente pesadas. Este tipo de infraestructuras solventa los problemas de computación masiva utilizando una gran cantidad de recursos agrupados en Organizaciones Virtuales (*VO*). En este sentido, la computación grid permite procesar, también, aplicaciones de uso intensivo de datos. Un ejemplo paradigmático de aplicaciones llevadas a cabo utilizando esta solución es el conjunto de experimentos desarrollados en el Gran Colisionador de Hadrones (*LHC*). Sin embargo, la naturaleza dinámica de este tipo de infraestructuras junto con las características heterogéneas y cambiantes de los recursos grid dan lugar a limitaciones en la planificación de tareas. De hecho, esto se ha convertido en un reto difícil de resolver debido a que el rendimiento, la eficiencia y disponibilidad de los recursos varía con el tiempo, perjudicando el rendimiento y ejecución de las aplicaciones.

Una alternativa cada vez más utilizada para resolver esta problemática consiste en proporcionar una capacidad auto-adaptativa a las aplicaciones, de manera que, estos cambios del entorno puedan ser gestionados por las mismas. Como se verá en la sección II, son muchas las formas en que se puede dotar a las aplicaciones de esta adaptación (mediante frameworks, mejorando el sistema, con nuevas políticas de scheduling, etc). Sin embargo, en este estudio se proporciona esa capacidad desde el nivel de aplicación. Para ello, se define un modelo de selección eficiente que aplica la metodología *BD* (Búsqueda Dispersa) para escoger los recursos en base a unos criterios de calidad y diversidad.

El artículo está estructurado de la siguiente manera. En la Sección II se realiza una descripción sobre el estado del arte en el área de auto-adaptación grid. El modelo de selección eficiente *ERS* (Efficient Resources Selection) propuesto se expone en la Sección III. La fase de evaluación, donde se muestran los resultados obtenidos en una grid real, se presenta en la cuarta sección. Finalmente, la Sección V concluye el artículo.

II. Estado del Arte

Dentro de la comunidad grid, existen investigaciones que abordan el tema de la auto-adaptación para afrontar la naturaleza dinámica de dichos sistemas. En este sentido, el trabajo presentado en [3] propone una ejecución adaptativa en grid definiendo un framework, basado en GLOBUS, que gestione los trabajos de una forma más eficiente. Por su parte, en [4] la investigación se centra en mejorar los procesos de descubrimiento y monitorización utilizando el paradigma de la Computación Autonómica. Se propone también un nuevo Servicio de Información Grid (*GIS*). El estudio en [5] proporciona el diseño de un sistema grid que dinámicamente se va ajustando al paralelismo de la aplicación. De esta forma, el propio sistema maneja los cambios de la infraestructura. Dos políticas de planificación son descritas en el mismo: una se centra en la suspensión/cancelación de tareas y la otra en gestionar el proceso de migración de las mismas.

Otros trabajos investigan la capacidad que puedan tener las propias aplicaciones para adaptarse. En [6], se expone una metodología de selección de recursos, donde la idea principal es que durante la ejecución

[1]Dpto. de Ciencia y Tecnología, Ceta-Ciemat, e-mail: {maria.boton, francisco.prieto}@ciemat.es

[2]Dpto. de Tecnología de Computadores y Comunicaciones, Univ. Extremadura, e-mail: mavega@unex.es

de la aplicación (y con cierta periodicidad) se recoja información relativa a la comunicación y tiempos de procesamiento de los recursos. En base a esta información, el conjunto de recursos con el que se trabaja se va ajustando, ya sea añadiendo o eliminando nodos. El objetivo del trabajo es resolver los problemas de cuello de botella y de sobrecarga de recursos, apreciados en estos entornos, a la vez que se mejora su rendimiento. Un estudio similar se presenta en [7], donde se describe una alternativa que maneje las aplicaciones grid autónomamente. Esta propuesta usa tanto el estado interno de la infraestructura como cierto conocimiento que obtiene a través de sensores externos para seleccionar el conjunto de recursos a utilizar. Se introduce, además, el concepto de simulación *viva* que implica, entre otras cosas, realizar procesos de migración de la aplicación de una máquina a otra, utilizando la información recabada. Finalmente, un informe de sistemas adaptativos capaces de manejar las condiciones cambiantes de la grid, mediante la toma de decisiones autónomamente se incluye en [8].

Los diferentes estudios mencionados anteriormente tienen un propósito común: mejorar el rendimiento de la infraestructura presentando soluciones a nivel de administración y/o sistema (técnicas de planificación, evitar cuellos de botella, reducir la carga de trabajo de los recursos, etc). Sin embargo, el presente trabajo propone un modelo de selección eficiente de recursos *ERS* basado en mejorar la productividad de la grid, proporcionando una capacidad auto-adaptativa a las aplicaciones. Para ello, el modelo es diseñado desde el punto de vista del usuario, lo que implica que no existe un control de los componentes internos del entorno. Esta es una diferencia significativa con respecto a los trabajos mencionados previamente: nuestro modelo simplemente elige los recursos en base a una serie de métricas de eficiencia, pero sin cambiar ni las características ni el comportamiento de los elementos grid.

III. Gestión Eficiente Basada en la Búsqueda Dispersa

La gestión eficiente de recursos basada en la búsqueda dispersa aplica el modelo *ERS* para determinar los recursos que intervienen durante la ejecución de la aplicación. A esta versión del modelo se le ha denotado como *ERS-SS* (Efficient Resources Selection based on Scatter Search). Esta estrategia está compuesta por los siguientes mecanismos biológicos evolutivos: *mutación, recombinación y selección.* Antes de proseguir con la descripción del modelo se van a dar unas nociones breves de grid para entender aún mejor el funcionamiento del mismo.

Las infraestructuras grid se dividen en 4 capas principales: la capa más baja es conocida como **capa de red** y es la encargada de conectar los recursos grid. Sobre ésta se encuentra la **capa de recursos** que contiene los elementos computacionales, sistemas de almacenamiento, catálogos de datos digitales, etc (la Tabla 1 presenta un resumen de los mismos). La capa intermedia en terminología grid se conoce como **middleware** y proporciona las herramientas que permiten que los recursos grid trabajen como un todo. Finalmente, en lo más alto se ubica la **capa de aplicación**, donde están las aplicaciones científicas, frameworks y herramientas de desarrollo. En esta capa se ubica el *ERS-SS*, como si de una subcapa de la misma se tratase.

TABLA I
Elementos básicos en una infraestructura grid

Acrónimo	Elemento	Función
UI	Interfaz de usuario	Punto de acceso a la infraestructura.
CE	Elemento de computación	Planificador que gestiona la cola de tareas en un site.
WN	Nodo de cómputo	Elemento donde se ejecutan las tareas. Es gestionado por los CEs en cada site.
SE	Elemento de almacenamiento	Elemento con capacidad de almacenamiento.
RB	Gestor de recursos	Metaplanifinicador que maneja el balance de carga y gestiona el almacenamiento grid.
BDII	Sistema de información global	Sistema de información global que mantiene el estado y la información de los recursos.
LFC	Catálogo de ficheros	Catálogo de ficheros distribuido que gestiona un espacio global de nombres.

A. Búsqueda Dispersa

La Búsqueda Dispersa (*BD*) [9]-[12] es un método evolutivo que ha sido utilizado para resolver un gran número de problemas de optimización. La primera descripción de este método fue realizada por Fred Glover [9] en 1977, el cual establece los principios fundamentales del método. La *BD* se basa principalmente en combinar soluciones para crear otras nuevas mejoradas. Esta combinación se realiza aplicando una serie de elecciones sistemáticas y estratégicas sobre el conjunto referencia (conjunto relativamente pequeño de soluciones). Hay que destacar que este conjunto tiene las soluciones buenas que se han ido encontrando durante la búsqueda. Para determinar si una solución es buena no hay que restringirse únicamente al concepto de calidad, sino también con-

siderar la diversidad que esta solución aporta al conjunto. A continuación se especifican los elementos que forman parte de la versión básica de este método:

- **Generador de soluciones diversas.** El conjunto G de soluciones diversas resultante se utiliza para obtener el conjunto referencia Rf.
- **Conjunto de referencia.** Debe contener soluciones buenas que sean diferentes entre sí (con cierto nivel de calidad y diversidad). Por lo general, Rf tiene la mitad de sus soluciones de *calidad* y la otra de *diversidad*. Todas las soluciones están ordenadas por calidad.
- **Método de combinación.** Se utilizan subconjuntos de 2 o más elementos $S \subset Rf$ que se combinan mediante una estrategia diseñada para tal efecto.
- **Método de mejora.** Por lo general, en esta parte se suele utilizar un método de búsqueda local para mejorar las soluciones.

Para finalizar, en la Figura 1 se muestran los pasos principales que conforman la ejecución del algoritmo. Notar que éste finaliza si al intentar realizar una combinación no existen nuevos elementos en el conjunto referencia.

BÚSQUEDA DISPERSA

1. Generar conjunto de soluciones diversas **G**;
2. Construir conjunto de referencia **Rf**;
   2.1 Mitad con mejores soluciones y otra mitad con las soluciones de **G** más diversas a las ya incluidas;
3. Generar los subconjuntos de **Rf**;
4. **Mientras** queden subconjuntos sin examinar **hacer**
   4.1 Seleccionar un conjunto **S** y aplicar el **método de combinación** a sus soluciones;
   4.2 Aplicar **método de mejora** a cada solución obtenida. Si la solución resultante mejora a **Rf**: sustituir y reordenar;
5. **Fin mientras**
6. Si entró alguna solución en **Rf** volver al paso 3, en otro caso **parar**;

Fig. 1. Esquema que muestra los pasos básicos de la Búsqueda Dispersa.

B. *Selección Eficiente de Recursos Grid. Facilitando la Auto-adaptación de Aplicaciones*

El modelo *ERS* está basado en el mapeo entre dos espacios: por un lado, un espacio de jobs J compuesto por las n tareas independientes y paralelas de la aplicación. Por otro lado, un espacio de recursos R que incluye los m elementos heterogéneos pertenecientes a la infraestructura grid en uso. En concreto, los recursos de este conjunto representan a los *CEs* del entorno. Para cada tarea t_α se escoge un recurso $Rt_\alpha \subset R$ de manera eficiente. Además, la asignación *tareas-recursos* es una relación de tipo *muchas-a-uno*.

Antes de describir el comportamiento del modelo basado en *BD*, se especifican los elementos que van a tomar parte en el mismo:

- **Conjunto calidad** C. Formarán parte de él los 5 mejores recursos a nivel de eficiencia.
- **Conjunto diversidad** D. Serán los 5 recursos más diversos respecto de los del conjunto calidad. Para determinar el valor de diversidad de un recurso se tienen en cuenta tanto criterios geográficos como de eficiencia (Figura 2). Cada recurso lleva asociado el identificador numérico del site al que pertenece SRt_α. Entonces, el valor de diversidad va a indicar cómo de diferente es un recurso, geográficamente hablando, respecto al conjunto calidad. Dicho valor se calcula para todos aquellos elementos que no formen parte del primer conjunto. Después, se ordenan de mayor a menor valor de diversidad y se escogen los 5 primeros que superen un umbral de fitness U establecido.
- **Umbral de fitness** U. Valor medio del fitness de todos los recursos evaluados en un cierto instante de tiempo. Será actualizado cada vez que un recurso finaliza una tarea.

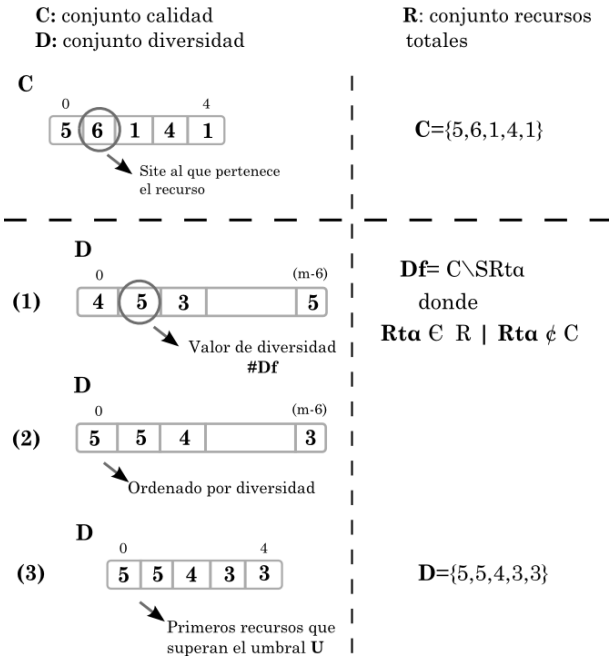

Fig. 2. Diagrama donde se reflejan los pasos para crear los conjuntos calidad y diversidad que utiliza el modelo *ERS*.

En cuanto al flujo de ejecución del modelo, los pasos principales (Figura 3) son descritos a continuación. Una vez que se ha preparado el entorno de la aplicación, se escoge un conjunto de tareas $P_\alpha \subset J$ que será lanzado inicialmente. Para este conjunto de tareas se seleccionan aleatoriamente una serie de recursos en los que ser ejecutadas (en este momento no se tienen métricas de eficiencia). Cabe destacar que cada tarea lanzada tiene un *tiempo de vida lt*, es decir, el tiempo que tiene el recurso asociado para procesarla. Cuando este *tiempo de vida* finaliza o la tarea es procesada, las métricas de eficiencia relativas al recurso Rt_α son actualizadas. Una vez se actualizan estas métricas y se recalculan los respectivos conjuntos, se selecciona un recurso de manera eficiente para una nueva tarea. Existe una condición de selección según la cual el nuevo recurso debe

pertenecer al conjunto de procedencia del Rt_α que ha acabado (si el recurso que termina pertenecía al conjunto calidad, el nuevo recurso será escogido de este mismo, y si pertenecía al conjunto diversidad, el nuevo recurso será escogido de este último). Esta alternancia entre conjuntos va a permitir que no sólo se usen los recursos más eficientes (propios del conjunto calidad) sino que se exploren otros nuevos que pueden llegar a ser tan eficientes como los anteriores (propios del conjunto diversidad). Finalmente, estos procesos de selección, ejecución y monitorización se repiten hasta que se procesan todas las tareas de J. Cuando la ejecución de la aplicación finaliza, la información generada durante todo el proceso es registrada en unos ficheros de salida.

PSEUDOCODIGO ERS-SS

Input: tareas de la aplicación, recursos de la infraestructura
Output: conjunto de soluciones

1. Preparar el entorno, generar aleatoriamente los conjuntos **C** y **D**;
2. Se lanza un subconjunto de tareas P_α;
3. **mientras** existan tareas por ejecutar **hacer**
   3.1 Por cada tarea comprobar tiempo de vida (lt) y estado;
      3.1.2 **Si** lt=0 o estado=DONE **entonces**;
         3.1.2.1 Calcular eficiencia y fitness del recurso Rt_α;
         3.1.2.2 Actualizar umbral de fitness (U);
         3.1.2.3 Obtener nuevos conjuntos **C** y **D**;
         3.1.2.4 Escoger un recurso del conjunto al que pertenecía Rt_α;
         3.1.2.5 Lanzar una nueva tarea;
4. **Fin mientras**

Fig. 3. Principales reglas que gobiernan la versión *ERS-SS*.

Para obtener el fitness de un determinado recurso i, lo primero es calcular su *tiempo de procesamiento* T_i como se muestra en Ec.1, donde $Tcom_i$ es el tiempo de comunicación entre el recurso CE_i y otros servicios grid y $Tcomp_{j,i}$ es el tiempo de procesamiento de la tarea j en dicho recurso.

$$T_i = Tcom_i + Tcomp_{j,i}\,. \tag{1}$$

A continuación, se determina el tiempo consumido por este recurso respecto al *tiempo de vida* asignado a la tarea (Ec.2). A este tiempo se le denota como θ_i; cuanto más próximo a 1 sea su valor indicará que más rápidamente ha procesado el recurso CE_i a la tarea j.

$$\theta_i = (lt - T_i)/lt\,. \tag{2}$$

El fitness F_i es calculado utilizando este tiempo θ_i junto con otros tres parámetros como se observa en Ec.3. Por un lado, ϵ_i es el porcentaje de éxito o fallo que obtiene el recurso para la ejecución de la tarea j. Por otro lado, se tienen dos *parámetros de relevancia* a y b que indican el peso relativo para ϵ_i y θ_i respectivamente. De este modo, los usuarios pueden determinar las condiciones de prioridad de sus experimentos (ya que el valor de los mismos es indicado por ellos).

$$F_i = (a \cdot \epsilon_i + b \cdot \theta_i)/(a + b) \tag{3}$$

Finalmente, aplicando estas métricas y empleando la técnica *BD* el modelo es capaz de descartar aquellos recursos que no están rindiendo de forma adecuada para la aplicación, ya sea por estar sobrecargados, por no estar disponibles o por cualquier otra circunstancia. No olvidemos que el objetivo es definir un modelo en el nivel de aplicación que guíe a las aplicaciones durante su ejecución en los entornos grid.

IV. Experimentación y Resultados

Para la evaluación del modelo se han definido dos escenarios, de manera que se pueda comprobar que se alcanzan los dos objetivos siguientes: *reducción del tiempo de ejecución de la aplicación e incremento de la tasa de tareas finalizadas con éxito*. El modelo ha sido probado en una infraestructura grid real con un número relativamente amplio de *CEs*. Esta infraestructura de pruebas pertenece al proyecto conocido como *Iniciativa Grid Nacional* (*NGI*) para España [13]. Esta iniciativa *NGI* española tiene como finalidad coordinar y promover el desarrollo de la actividad científica a través del uso colaborativo de recursos geográficamente distribuidos. Está formada por un número significativo de *VOs*, la mayoría de ellas creadas para un determinado proyecto o para trabajar en relación a un tópico concreto (existen tres *VOs* de naturaleza genérica). De este rango de organizaciones, hemos escogido para la fase de evaluación de nuestro modelo aquella que pertenece al proyecto IBERGRID [14] (*iber.vo.ibergrid.eu*). Por otra parte, la aplicación utilizada en los diferentes escenarios se basa en el método Runge-Kutta [15], un conocido procedimiento para resolver problemas de ecuaciones diferenciales de manera numérica. Cabe destacar que en la etapa de diseño del modelo nos hemos centrado en dotar de una capacidad adaptativa a aplicaciones paralelas, donde no existe comunicación entre las diferentes tareas. Finalmente, para verificar que el modelo consigue realizar una selección eficiente de los recursos con *ERS-SS*, se compara dicha versión con la selección tradicional en grid (denotada como *TRS*). Los criterios de selección de recursos en *TRS* se basan principalmente el la disponibilidad y cercanía de los recursos así como en los requisitos de las tareas, especificados por los usuarios.

A. Escenario 1: Aprendizaje del Modelo

En este primer escenario el objetivo es determinar la influencia del tamaño del conjunto P_α (jobs lanzados a ejecución la primera vez en cada prueba) del modelo sobre la ejecución de la aplicación. En otras palabras, se trata de obtener el tamaño mínimo de ese conjunto (aprendizaje mínimo) para alcanzar una reducción de tiempo significativa con respecto a *TRS*. Para ello, se han definido 5 tests, cada uno con un espacio J de 200 tareas. Además, en cada test el número de iteraciones del modelo varía y, consecuentemente, el tamaño del conjunto de tareas P_α que se lanza inicialmente a ejecución también cambia (ver Tabla II). Los valores de los parámetros a y b se fijan a 60% y 40% respectivamente. Hay que que

indicar que, en ambos escenarios, cada punto de la gráfica es el valor medio aritmético obtenido a partir de 10 repeticiones del experimento (esto es así para cada modelo de selección de recursos).

TABLA II
CARACTERÍSTICAS DEL PRIMER ESCENARIO DE EVALUACIÓN. SE PRETENDE DETERMINAR CÓMO INFLUYE EL APRENDIZAJE DEL MODELO.

Test	Tamaño P_α	Tamaño J	Iteraciones
1	5	200	40
2	10	200	20
3	13	200	15
4	20	200	10
5	40	200	5

A partir de los resultados obtenidos (Figura 4) puede deducirse que el modelo propuesto mejora con mucha diferencia a la selección tradicional en grid. Hay que tener en cuenta que el tiempo de ejecución especificado en el caso de *ERS-SS* incluye no sólo el tiempo de ejecución de la aplicación sino también el tiempo de comunicación entre el modelo y la aplicación así como el tiempo usado por el modelo para monitorizar y clasificar los recursos. Se aprecia que en las primeras pruebas de este escenario es cuando el modelo alcanza sus valores mínimos, ya que el número de tareas que se lanzan en un principio es mayor (provocando que el aprendizaje sea más rápido).

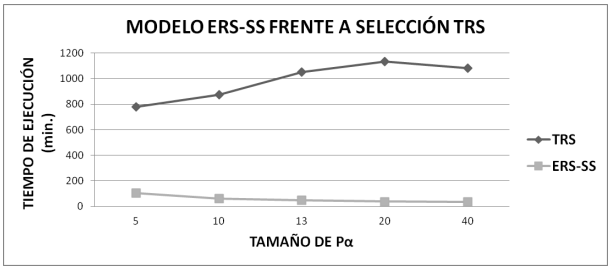

Fig. 4. Tiempos de ejecución obtenidos para los distintos niveles de aprendizaje del modelo. Puede apreciarse que la versión *ERS-SS* alcanza mejores valores.

Pero, no sólo se alcanza el primero de los objetivos marcados en esta etapa, sino que también se produce un incremento en la tasa de tareas finalizadas con éxito, como se observa en la Figura 5. En el caso del *TRS* esta tasa es de un 73% mientras que el ERS-SS llega a una tasa de éxito del 96%.

B. Escenario 2: Rango de aplicaciones

El objetivo de este segundo escenario es calcular el efecto del tamaño del espacio J (total de jobs a ejecutar por la aplicación) en el rendimiento de la aplicación al utilizar el modelo de selección propuesto. De esta forma sabremos el rango de aplicaciones donde es factible aplicar nuestra estrategia inteligente. Por ello, el tamaño de J irá variando en los diferentes tests como puede observarse en la

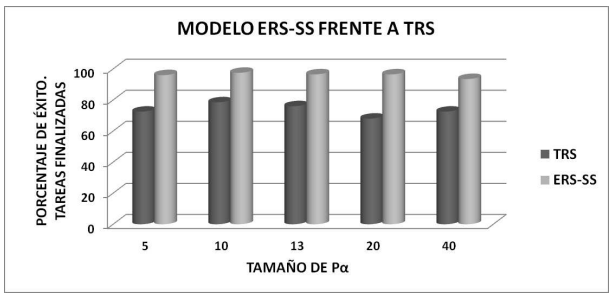

Fig. 5. Tasas de éxito obtenidas para los dos modelos. El modelo *ERS-SS* supera en todos los casos la tasa del TRS.

Tabla III. Además, en todas las pruebas se lanzarán inicialmente 10 tareas (valor de P_α). Se supone que cuanto mayor sea el tamaño de J, mayor diferencia en tiempo de ejecución se apreciará entre *ERS-SS* y *TRS*, ya que el modelo tendrá un aprendizaje mayor además de sacar un mayor beneficio de dicho modelo inteligente (por ejecutarse ambos un mayor número de veces). Por este motivo, intuimos que *ERS* rendirá mejor para grandes producciones.

TABLA III
VALORES ESTABLECIDOS EN EL SEGUNDO ESCENARIO PARA DETERMINAR EL RANGO DE APLICACIONES DONDE APLICARLO.

Test	Tamaño J	Tamaño P_α	Iteraciones
1	50	10	5
2	100	10	10
3	200	10	20
4	300	10	30
5	400	10	40
6	500	10	50

Puede observarse en la Figura 6 que nuevamente el modelo *ERS-SS* alcanza mejores resultados de tiempo que la selección propia de las infraestructuras grid. Por ejemplo, para el caso de un tamaño de 500 tareas, la estrategia *TRS* tarda más de dos días (3019 minutos) mientras que *ERS-SS* consume tan sólo 108 minutos (menos de dos horas). Tal y como se ha supuesto en un principio, a medida que el tamaño del espacio de tareas aumenta lo hace también esa diferencia de tiempos.

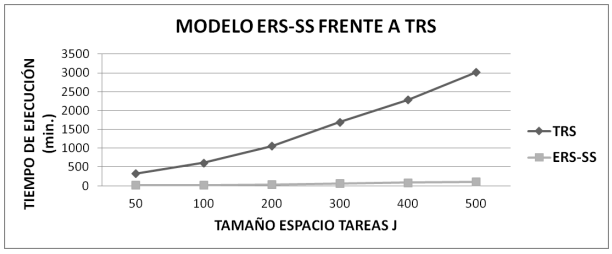

Fig. 6. Gráfica que muestra los resultados obtenidos para *ERS-SS* y *TRS* respectivamente.

Al igual que ocurría en el escenario anterior, no sólo se consiguen mejores resultados en los tiempos de ejecución, sino que el modelo propuesto garan-

tiza, además, una tasa media de éxito (en la ejecución de las tareas) del 96% en los 6 tests realizados (Figura 7).

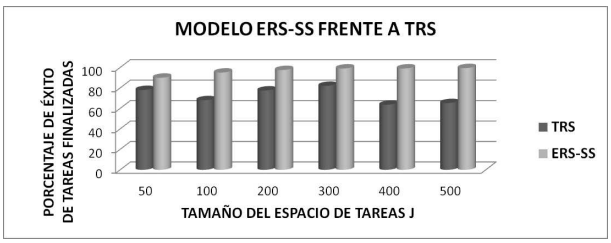

Fig. 7. Tasa de éxito en este segundo escenario para los dos modelos empleados.

Finalmente, a partir de los datos obtenidos en las dos fases de evaluación se puede concluir que el modelo cumple los objetivos marcados (*reducir el tiempo de ejecución de la aplicación e incrementar la tasa de tareas finalizadas con éxito*) y que puede ser utilizado en aplicaciones grid de grandes producciones. En definitiva, se hace un uso eficiente de los recursos grid, mejorando la productividad de la infraestructura y permitiendo a las aplicaciones adaptarse a los cambios.

V. Conclusiones y Trabajo Futuro

El presente trabajo se centra en resolver el problema de selección de recursos grid proporcionando una capacidad auto-adaptativa a las aplicaciones. Para llevar a cabo esta estrategia inteligente, se mide el fitness y la eficiencia de los recursos de forma constante. El fitness de un recurso se calcula en base al éxito en la ejecución de sus tareas y a su tiempo de procesamiento. La selección de recursos eficiente y auto-adaptativa en los entornos de computación grid es una solución factible que asegura la adaptación de las aplicaciones a los cambios de la infraestructura, mejorando a su vez la productividad de la misma. Esto se puede observar en los resultados obtenidos durante la fase de evaluación del modelo. La versión desarrollada *ERS-SS* es comparada con la selección tradicional en grid. Se consiguen mejoras significativas utilizando nuestro modelo propuesto.

Como trabajo futuro, se estudiarán otros algoritmos para incluirlos en el modelo y se mejorarán las funcionalidades de los diferentes módulos que componen *ERS*. Además, se considerarán otros servicios grid y otras circunstancias que influyan negativamente en la ejecución de las aplicaciones.

Agradecimientos

La investigación de María Botón-Fernández está subvencionada mediante una beca de Formación de Personal Investigador otorgada por el Ministerio de Economía y Competitividad a través del Centro de Investigaciones Energéticas, Medio Ambientales y Tecnológicas (CIEMAT). Los autores quieren agradecer también el soporte de los Fondos Europeos para el Desarrollo Regional (FEDER).

Referencias

[1] I. Foster, *What is the Grid? A Three Point Checklist*, in: GRIDtoday, Vol. 1, No. 6, pp. 22-25, 2002.

[2] I. Foster, C. Kesselman, S. Tuecke, *The Anatomy of the Grid. Enabling Scalable Virtual Organizations*, in: R. Sakellariou, J.A. Keane, J.R. Gurd, L. Freeman (eds.), Euro-Par 2001, LNCS, Vol. 2150, Springer-Verlag, Heidelberg, pp. 1-4, 2001.

[3] E. Huedo, R.S. Montero and I.M. Llorente, *A Framework for Adaptive Execution in Grids*, in: Software-Practice & Experience, Vol. 34, Issue 7, John Wiley and Sons Inc., New York, NY, USA, pp. 631-651, 2004.

[4] H.N.L.C Keung, J.R.D. Dyson, S.A. Jarvis and G.R. Nudd, *Self-adaptive and Self-optimising Resource Monitoring for Dynamic Grid Environments*, in: DEXA'04 Proceedings of the Database and Expert Systems Applications, 15th International Workshop, IEEE Computer Society, Zaragoza, Spain, pp. 689-693, 2004.

[5] S.S. Vadhiyar and J.J. Dongarra, *Self Adaptivity in Grid Computing*, in: Concurrency and Computation: Practice & Experience, Vol. 17, Issue 2-4, John Wiley and Sons Ltd., Chichester, UK, pp. 235-257, 2005.

[6] G. Wrzesinska, J. Maasen and H.E. Bal, *Self-adaptive Applications on the Grid*, in: 12th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programing, San Jose, California, USA, pp. 121-129, 2007.

[7] D. Groen, S. Harftst and S.P. Zwart, *On the Origin of Grid Species: The Living Application*, in: Proceedings of the 9th International Conference on Computational Science: Part I, LNCS, Vol. 5544, Springer-Verlag, Heidelberg, pp. 205-212, 2009.

[8] D.M. Batista and L.S. da Fonseca, *A Survey of Self-adaptive Grids*, in: IEEE Communications Magazine, Vol. 48, Issue 7, IEEE Press Piscataway, NJ, USA, pp. 94-100, 2010.

[9] F. Glover, *Heuristics for Integer Programming Using Surrogate Constraints*, in: Decision Sciences, Vol. 8, Decision Sciences Institute, pp. 156-166, 1977.

[10] M. Laguna and R. Martí, *Scatter Search: Methodology and Implementations in C*, in: Kluwer Academic Publishers, 2003.

[11] M. Laguna and R. Martí, *Scatter Search*, in: E. Alba, R. Martí (eds.), Metaheuristic Procedures for Training Neural Networks, Springer, pp. 139-152, 2006.

[12] M. Resende, C. Ribeiro, F. Glover, R. Martí, *Scatter Search and Path Relinking: fundamentals, Advances and Applications*, in: M. Gendreau and J.Y. Potvin (eds.), Handbook of Metaheuristics, Springer, 2009.

[13] Página Web de la Iniciativa Grid Nacional en España, `http://www.es-ngi.es/`

[14] Página Web del Proyecto Ibergrid, `http://www.ibergrid.eu/`

[15] W.H. Press, B.P. Flannery, S.A. Teukolsky and W.T. Vetterling, *Numerical Recipies in C*, in: Press syndicate of the University of Cambridge, New York, 1992.