Una Herramienta de Programación Genética Paralela que Aprovecha Recursos Públicos de Computación

F. Chávez[1], J. L. Guisado[1], D. Lombraña[1], F. Fernández[1].

Resumen--**Éste artículo presenta una primera implementación de una herramienta genérica de programación genética capaz de aprovechar recursos públicos de computación. Dadas las altas necesidades de recursos de computación requeridos por los algoritmos evolutivos, la aplicación del paralelismo ha sido habitual recientemente, aunque las herramientas paralelas requieren infraestructuras costosas para su aprovechamiento. El modelo que se presenta en este artículo, permite utilizar computadores distribuidos en Internet, cuyos usuarios ceden altruistamente para colaborar en proyectos de investigación. El proceso de donación de recursos es simple e inmediato por parte del usuario, afectando solamente a los ciclos de CPU que no son consumidos por el propio usuario. Se estudia la mejora de las prestaciones obtenidas gracias al uso de estos recursos en Programación Genética Distribuida.**

Palabras clave—**Algoritmos Paralelos, Programación Genética.**

I. INTRODUCCIÓN.

En los últimos años, se han presentado herramientas que permiten ejecutar diferentes tipos de algoritmos evolutivos en paralelo. La idea es aprovechar infraestructuras de computación paralelas, para ganar tiempo en la búsqueda de soluciones cuando se utilizan este tipo de metaheurísticas. Entre estos trabajos, cabe citar la herramienta ECJ [1] y el proyecto DREAM[2].

Este tipo de herramientas están pensadas para que los usuarios/científicos puedan ejecutarlas en las infraestructuras paralelas de las que disponen, ya sean máquinas multiprocesador o clusters de computadoras, y a la vez poder colaborar con otros investigadores que también disponen de infraestructuras similares.

La computación GRID [3, 10] permite avanzar un paso más en estos modelos de computación, ofreciendo la posibilidad de unificar de manera coherente todas esas infraestructuras para que su utilización pueda realizarse de manera homogénea. ParadisEO [13], por ejemplo, es una herramienta paralela para Algoritmos Evolutivos que se ha utilizado de forma satisfactoria en entornos GRID.

Pero además de los modelos anteriores, existe un concepto que ha sido aprovechado con éxito en proyectos de investigación internacionales con alta demanda computacional: la utilización de *recursos públicos de computación (PRC).* El proyecto SETI [4] promovió el uso de ciclos de CPU que usuarios diseminados a través de Internet pudieran ceder a un proyecto de interés. El éxito fue rotundo, consiguiendo prestaciones comparables a las de los grandes supercomputadores.

Como resultado del proyecto SETI, se liberó la tecnología BOINC [9] que permite adaptar cualquier aplicación para que pueda aprovechar ciclos de CPU de usuarios particulares que quieran colaborar.

Este trabajo presenta, hasta donde sabemos, la primera aplicación de la tecnología BOINC al dominio de los algoritmos evolutivos. En particular presentamos una primera adaptación de la herramienta [12] para que pueda aprovechar este tipo de recursos.

El resto del artículo se estructura del siguiente modo: En las secciones 2 y 3 describimos la metodología utilizada. En las secciones 4 mostramos la nueva herramienta y su forma de uso. En la sección 5 mostramos los resultados obtenidos, y finalmente en la sección 6 presentamos las conclusiones del estudio.

II. PROGRAMACIÓN GENÉTICA PARALELA Y PRC.

La Programación Genética (PG) ha sido paralelizada en los últimos años de manera frecuente. Como en el resto de Algoritmos Evolutivos, se trata siempre de ahorrar tiempo utilizando un número alto de procesadores.

En particular, el modelo de islas ha sido aplicado con éxito [5], y son conocidas las ventajas que aporta tanto a la búsqueda de soluciones, como en el control del tamaño de los individuos [6]. Herramientas como la descrita en [7, 1] son adecuadas cuando se utiliza un sistema multiprocesador o cluster. Sin embargo, en ocasiones, no se dispone de infraestructura de este tipo.

Son pocos los trabajos que han explorado el área de los Recursos Públicos de Computación (PRC) aplicados a la PG. En [8] se describe una implementación específica que pretende la parametrización de las ejecuciones en PG para

[1] Universidad de Extremadura. Departamento de Informática. Dirección del autor 1. E-mail: fchavez@unex.es

llegar a conclusiones sobre el comportamiento y rendimiento de esta heurística.

Sin embargo, la idea de utilizar de modo sistemático PRC con algoritmos evolutivos no ha sido descrita hasta el momento.

Entre las diferentes posibilidades existentes, en este trabajo presentamos una adaptación de la herramienta clásica de PG LilGP [12] para aprovechar ciclos de CPU donados por usuarios particulares a través de Internet.

Para ello hemos seleccionado el esqueleto suministrado por BOINC [5]. La implementación permite utilizar la herramienta en cualquier problema deseado, y requiere para su utilización un computador que actúe como servidor del "proyecto". A continuación se describe la arquitectura BOINC y su aplicación a la herramienta LilGP.

III. LA INFRAESTRUCTURA BOINC.

BOINC (Berkeley Open Infrastructure for Network Computing) es una infraestructura de software para realizar computación distribuida mediante el paradigma "Desktop Grid", es decir, utilizando un conjunto heterogéneo de computadores interconectados a través de una red del tipo Internet (ya sean computadores independientes o un conjunto de ellos pertenecientes a una determinada organización).

La unidad básica sobre la que se estructura la computación es el *proyecto BOINC*. Consiste en un grupo de una o más aplicaciones distribuidas que utilizan la plataforma BOINC, manejadas por una determinada organización. Cada proyecto BOINC se identifica por una única URL maestra, que es la página de inicio de su sitio web. Los usuarios que deseen colaborar con el proyecto, donando ciclos de reloj, deberán unirse al proyecto registrándose a través de la URL maestra del mismo. Posteriormente, estos usuarios pueden contribuir a la ejecución de las herramientas servidas por el proyecto con tantos equipos como deseen, por medio de la ejecución en sus máquinas locales del cliente BOINC. Será tarea del investigador, hacer el proyecto públicamente atractivo para conseguir usuarios que colaboren.

La infraestructura BOINC se compone de una parte servidora y otra parte cliente (ver Fig. 1). La parte servidora, que puede residir en un único servidor o en varios (proporcionando cada uno parte de la funcionalidad), consta de una base de datos relacional y un conjunto de servicios web y procesos demonio: *servidores de distribución de tareas, servidores de datos y servidor web*.

Los distintos componentes de la infraestructura BOINC y la interrelación entre los mismos se muestran en la Fig. 2. Los componentes que se muestran en tono oscuro en dicha figura son proporcionados como parte del sistema BOINC, mientras que el resto deben ser desarrollados por los gestores de la aplicación que se pretenda ejecutar.

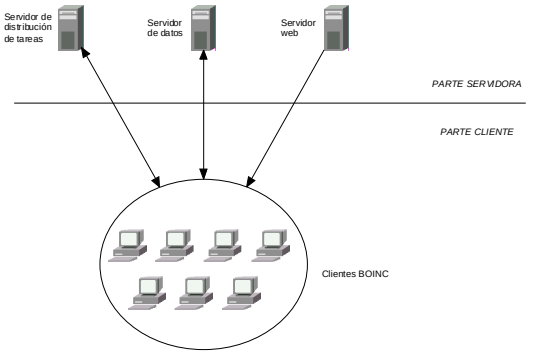

Fig. 1. Visión a gran escala de la infraestructura BOINC mostrando sus elementos básicos.

En la parte servidora, la *base de datos de BOINC* se encarga de almacenar descripciones de todos los componentes que intervienen en cada uno de los proyectos BOINC que el servidor puede servir: aplicaciones, plataformas, versiones, resultados, cuentas, equipos, etc. El *servidor (o servidores) de distribución de tareas* realiza el envío de trabajos a cada uno de los clientes y la recepción de informes acerca de los trabajos ya completados por ellos. El *servidor (o servidores) de datos* maneja el envío a los clientes de la/s aplicaciones a ejecutar relativas al proyecto al cual se ha unido el usuario, junto con los archivos de datos de entrada para la ejecución de los trabajos. De igual forma, se encarga de la recepción de archivos de datos con los resultados procedentes de los trabajos completados por los clientes. Un *servidor web* se encarga de proporcionar la interfaz del proyecto hacia los clientes. Incluye una serie de páginas web estándar del sistema BOINC más otras páginas web dependientes de la aplicación que deben ser creadas por los gestores de la misma. Estas últimas pueden almacenar datos propios de la aplicación en una *base de datos de la aplicación*. Todos los componentes de la parte servidora son controlados por el software motor de BOINC, que puede estar complementado por un motor de la aplicación para controlar funciones específicas de la misma. Como se ha mencionado, todos estos servidores pueden estar situados físicamente en la misma máquina.

La parte cliente contiene el software a instalar en cada computador cliente para realizar la recepción de trabajos y archivos de entrada desde los servidores del proyecto, la ejecución de dichos trabajos y el envío a los servidores de los archivos de datos con los resultados. Consta de un *programa cliente BOINC* proporcionado por el

sistema y de un *programa cliente de la aplicación*, que debe ser desarrollado por los gestores de la misma para realizar la computación propia de la aplicación en cada computador cliente. El programa cliente de la aplicación se comunica con el cliente BOINC usando la API del sistema BOINC.

IV. Implementación y uso de LilGP & BOINC.

Para realizar la implementación de una herramienta de PG utilizando BOINC se ha seleccionado LilGP por su amplia difusión, y estructura fácil de adaptar a BOINC.

LilGP es una herramienta de PG escrita en lenguaje de programación C para el compilador gcc. Sin embargo, el servidor BOINC está desarrollado en C++ y se compila con g++. Las aplicaciones que un investigador desee boincificar y posteriormente ejecutar a través de un servidor BOINC deben seguir los estándares de C++ y tomar en cuenta las características de g++.

Hubo por tanto que realizar adaptaciones en el código, que afectaron a la declaración de las estructuras de datos y funciones del fichero para que pudiera ser compilado con g++.

Con los cambios detallados anteriormente, la herramienta puede ser compilada con g++, generando un ejecutable similar al anterior y ofreciendo los mismos resultados que la aplicación original.

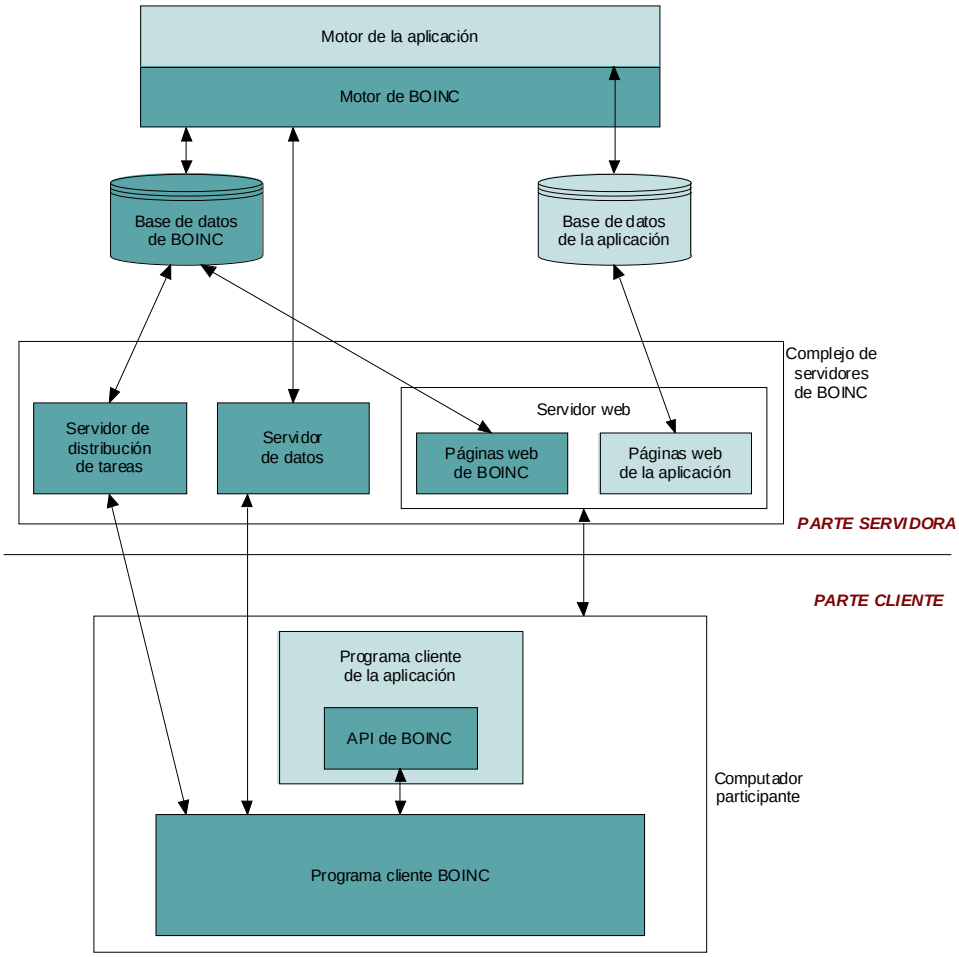

Fig. 2. Componentes de la infraestructura BOINC e interrelaciones entre los mismos. Aquellos con fondo oscuro son proporcionados como parte del sistema BOINC, mientras que aquellos con fondo claro deben ser desarrollados por los gestores de la aplicación que se pretenda ejecutar.

Además, tal como BOINC especifica, deben modificarse todos y cada uno de los accesos a los ficheros que se utilizarán a lo largo de la ejecución de la herramienta. La razón básica es que BOINC permite repetición de ejecuciones en las máquinas clientes, y es necesario el control de nombres de ficheros que se utilizan, para que no se produzcan escrituras indeseadas, tanto en la parte cliente, como en la parte servidora, que será la encargada de recoger todos los datos generados por los clientes en sus ejecuciones.

Dado que LilGP genera datos abundantes durante su ejecución, correspondientes a la evolución de las soluciones, hubo que modificar el código fuente utilizando las nuevas librerías de E/S suministradas por BOINC, y que evitan los problemas mencionados anteriormente.

Una vez realizado los cambios y compilada la herramienta, se decidió proceder con la utilización de la misma en un entorno controlado, que se describe en la siguiente sección, y utilizando un problema clásico en PG: el problema de la hormiga en la pista de Santa Fe [11].

En la herramienta LilGP, el problema de la hormiga viene integrado en la misma como ejemplo. Los únicos cambios que se efectuaron para la nueva versión fueron aquellos en los que intervenían operaciones de E/S sobre ficheros: lectura del fichero de entrenamiento para la hormiga. La salida de datos es generada por la propia herramienta LilGP, y las modificaciones necesarias habían sido realizadas en la herramienta básica.

Describimos a continuación la utilización de la nueva herramienta.

A. Utilización LilGP &BOINC.

Cualquier investigador interesado en utilizar la nueva versión de la herramienta para resolver un problema, deberá realizar las siguientes tareas:

0.- Descarga e instalación del servidor de proyectos BOINC, junto con los requisitos necesarios para su ejecución.

1.- Instalación de la herramienta LIPGP boincificada (disponible bajo petición a los autores del artículo).

2.- Implementación del problema a resolver bajo las especificaciones propias de la herramienta LilGP.

3.- Mantener la resolución de los ficheros utilizados por el nuevo problema bajo las premisas de BOINC, resolviendo las aperturas y cierres según sus especificaciones.

4.- Compilación de la herramienta para la generación del fichero ejecutable.

5.- Creación y configuración de un proyecto BOINC en el servidor, donde se subirá el fichero ejecutable generado por la herramienta para poder ser descargado y ejecutado por los clientes.

6.- Recogida de ficheros proporcionados por los clientes adheridos al proyecto, con el resultado de las generaciones ejecutas por ellos.

Para realizar una *prueba de concepto* se ha adaptado el problema de la hormiga en la pista de Santa Fe. Este problema es bien conocido en el dominio, y viene implementado en la herramienta LilGP original.

La adaptación del problema, ha consistido únicamente en la resolución de ficheros bajo las premisas de BOINC de la parte de la herramienta donde se codifica la resolución del problema (ver Fig. 3).

Resolución de ficheros estándar C/C++

```
#include "iostream"
...
int main() {
        …
    FILE *f;

    f=fopen(“nombre_fichero”, “w”);
    if (f = = null) {
        //Error
        return 0;
    }
    ...
    fclose(f);
    return 0;
}
```

Resolución de ficheros estándar BOINC

```
#include "BOINC/boinc_api.h"
#include "BOINC/boinc_zip.h"
#include "BOINC/filesys.h"
...
int main() {
        …
    FILE *f;
    int retval;
    char resolved_name = new char[80];

    boinc_init();

    retval = boinc_resolve_filename("nombre_fichero",
resolved_name, sizeof(resolved_name));
    if retval = 0 {
        //Error
        boinc_finish(retval);
    }
    f=boinc_fopen(resolver_name, “w”);
    ...
    fclose(f);
    boinc_finish(0);
}
```

Fig. 3: Ejemplo de Instrucciones de E/S propias de BOINC comparadas con C/C++.

Para que un usuario particular pueda colaborar en un proyecto basado en LilGP&BOINC, debe unirse al proyecto deseado a través de la web que genere éste. Para ello el usuario deberá descargar una herramienta cliente BOINC e instalarla en su computador. Una vez instalada, el usuario configurará la herramienta cliente con los datos relativos al proyecto al cual se ha unido, con ello podrá obtener tareas relativas al proyecto. A partir de ese momento su computador prestará ciclos para el proyecto automáticamente, retornando al servidor los ficheros con los resultados generados en las diferentes ejecuciones de la herramienta.

V. RESULTADOS EXPERIMENTALES Y MEDIDAS DE RENDIMIENTO.

Para realizar las pruebas hemos utilizado un PC Pentium IV, CPU 2.8 GHz, 512 MB de RAM y una tarjeta de red Gibabit Ethernet actuando como servidor BOINC. El sistema operativo instalado ha sido Debian 2.6.8 con APACHE 2.0 como servidor web y MySql como base de datos para gestión de proyectos, tareas y resultados de generaciones realizadas por los clientes.

Además, se ha configurado un aula de la Universidad de Extremadura, constituida por 10 PC Pentium IV, CPU 2.8 GHz,512 MB de RAM y una tarjeta Ethernet, con el sistema operativo Linex 2004. En estos equipos se ha instalado el cliente BOINC para poder efectuar las diferentes pruebas detalladas en las siguientes tablas.

Las pruebas han incluido un conjunto de experimentos con diferente número de generaciones y/o individuos en cada caso, cada uno de los cuales se repite 25 veces, como es habitualmente el caso cuando se trabaja con Algoritmos Evolutivos y se desean obtener resultados estadísticamente significativos. Se ha aprovechado la arquitectura BOINC para realizar las 25 ejecuciones de forma paralela. Solamente en una ocasión hemos repetido el experimento 5 veces, para estudiar el comportamiento con una sola tarea asignada a cada computador utilizado (ver primera fila de la tabla 1).

El problema PG utilizado es el problema de la hormiga en la pista de Santa Fe. Se han utilizado 5 y 10 máquinas actuando como cliente. Los resultados de las mismas se muestran en la Tabla 1 (5 máquinas cliente) y la Tabla 2 (10 máquinas cliente).

Tamaño	Nº de tareas	$t_{(secuencial)}$	$t_{(BOINC)}$	$a_{(BOINC)}$
50 G x 2000 I	5	130 s	307 s	0,0977
50 G x 2000 I	25	650 s	395 s	1,6456
1000 G x 1000 I	25	4250 s	1548 s	2,7455
2000 G x 1000 I	25	9200 s	2356 s	3,9049

Tabla 1. Resultados con 5 máquinas como cliente.

Tamaño	Nº de tareas	$t_{(secuencial)}$	$t_{(BOINC)}$	$a_{(BOINC)}$
1000 G x 1000 I	25	4250 s	1033 s	4,1142
2000 G x 1000 I	25	9200 s	1623 s	5,6685

Tabla 2. Resultados con 10 máquinas como cliente.

En las tablas anteriores se recogen los datos relativos al tamaño del problema que se ha ejecutado, indicando el número de generaciones del mismo (G), así como el número de individuos que intervienen en cada generación (I). Se muestra igualmente el número de tareas que efectúa cada máquina. Se ha medido el tiempo de CPU que sería necesario para la ejecución de las tareas en una máquina sin la infraestructura BOINC de forma secuencial ($t_{(secuencial)}$), el tiempo de CPU total empleado por las máquinas con infraestructura BOINC para la ejecución del mismo número de tareas ($t_{(BOINC)}$), y por último, la aceleración es calculada como:

$$a = \frac{t_{secuencial}}{t_{BOINC}}$$

No se ha estudiado la calidad de la solución de los problemas, porque el modelo de computación no afecta en absoluto a la calidad de las soluciones: se emplea el algoritmo básico de PG, y solamente se realiza en paralelo la repetición de tareas. Esta repetición de tareas es imprescindible cuando se analizan resultados de algoritmos con componente estocástica, para que al promediar resultados se obtengan datos estadísticamente significativos.

Se han considerado ejecuciones de diferentes tamaños, para mostrar la verdadera utilidad de la nueva herramienta basada en la arquitectura BOINC. Así, el número de tareas corresponde con el número de ejecuciones de un determinado experimento, mientras que el tamaño de las tareas ha dependido en cada caso del número de generaciones a evaluar (G) y el número de individuos de la población (I).

Aunque no ha sido el objeto de este estudio, hay que destacar que para el problema en particular utilizado para las pruebas, solamente en los casos de tareas costosas en tiempo, correspondientes a valores mayores de número de generaciones se ha encontrado la solución óptima. En algunas ocasiones, cuando la solución óptima se alcanza, la ejecución finalizará sin completar el máximo número de generaciones previsto.

Como se esperaba, debido a la arquitectura BOINC en la que los costes de comunicación son significativos, se obtienen buenos resultados cuando la tarea es más costosa. En la tabla 1 puede observarse que para tareas pequeñas se pierde aceleración debido a las comunicaciones, por lo que no estaría justificado el uso de la herramienta (ver el caso de 50 generaciones, población de 2000 individuos y repetición del experimento sólamente 5 veces).

En cambio, con un número de tareas más grande, (25 tareas del mismo tamaño) se consigue una aceleración significativa: 1.64. La aceleración no es tan grande como podría esperarse para 5 máquinas, pero los costes de comunicación son tan significativos, y el tamaño de las tareas tan pequeños que influyen notablemente en los resultados.

En la misma tabla puede observarse que si aumentamos el tamaño de la tarea (hasta 1000 o 2000 Generaciones y 1000 individuos), la aceleración sí empieza a ser relevante: 2,7 y 3,9 respectivamente.

Si en lugar de utilizar 5 clientes BOINC, utilizamos 10, las aceleraciones obtenidas para las tareas de mayor tamaño anteriores suben hasta 4,1 y 5,6 respectivamente. A mayor tamaño de tarea, y mayor número de clientes, las aceleraciones obtenidas son más importantes.

Hay que hacer notar de nuevo, que el investigador interesado en utilizar LilGP&BOINC solamente debe concentrarse en la instalación del servidor, con información detallada y atractiva del proyecto para conseguir donaciones sustanciosas de usuarios. El interés del proyecto será el que atraiga la colaboración de usuarios.

El modelo facilita la gestión de equipos: el investigador no necesita gestionar sistemas multiprocesadores ni configurar clusters para que trabajen de modo apropiado con una herramienta paralela. Es suficiente con configurar el servidor y abrirlo al mundo.

La herramienta LilGP&BOINC suministrada permite así a los investigadores implementar problemas que se desean resolver del mismo modo que con la herramienta clásica LilGP y permitiendo utilizar recursos públicos de computación de manera efectiva y cómoda.

VI. CONCLUSIONES.

En este artículo se ha descrito una adaptación de la herramienta de PG LilGP para su utilización distribuida aprovechando recursos públicos de computación.

La nueva versión de la herramienta se basa en el esquema BOINC, que permite la creación de proyectos centralizados, a los que usuarios particulares pueden contribuir de forma altruista, donando ciclos de reloj de sus computadores personales. Hasta donde sabemos, esta es la primera implementación de una herramienta genética de PG basada en BOINC que aproveche recursos públicos de computación.

Se ha realizado una prueba de concepto utilizando máquinas de un laboratorio de prácticas, disponible en la Universidad de Extremadura. Las pruebas, basadas en el problema de la hormiga de Santa fe, han mostrado la utilidad de la nueva implementación, consiguiendo ahorros de tiempo de cálculo muy significativos, y sin necesidad de disponer de grandes sistemas de computación, basados en máquinas multiprocesador o cluster de computadores.

Para las medidas de rendimiento hemos utilizado la plataforma BOINC para realizar computación institucional bajo el modo “Desktop Grid”, es decir, utilizando un conjunto homogéneo de computadores gestionados por una cierta institución. Sería interesante realizar como trabajo futuro la evaluación del rendimiento obtenido utilizando recursos públicos donados por voluntarios a través de Internet.

VII. REFERENCIAS

[1]. S. Luke et al, A Java-based Evolutionary Computation Research System http://cs.gmu.edu/~eclab/projects/ecj/

[2]. Arenas, M.G. Collet, P., Eiben, A. E., Jelasity, M., Merelo, J. J., Paechter, B., Preuß, M., Schoenauer, M., “A Framework for Distributed Evolutionary Algorithms”, Proceedings of PPSN VII, Granada, September 2002.

[3]. N. Melab, S. Cahon and E-G Talbi, “Grid Computing for parallel bioinspired algorithms”, in F Fernández and E. Cantú Paz (guest eds.) Special Issue on Parallel Bioinspired Algorithms, Journal of Parallel and Distributed Computing, Vol. 66, N. 8, pp. 1052-1061. Elsevier, Aug. 2006.

[4]. W. T. Sullivan D. Werthiner, S. Bowyer, J. Cobb, G. Gedye and D. Anderson. “A new major SETIproject based on Project Serendip data and 1000,000 personal computers.” In Proc. Of the Fith Intl. Conf. on Boiastronomy. 1997.

[5]. F. Fernández, M. Tomassini, L. Vanneschi, "An Empirical Study of Multipopulation Genetic Programming", . Genetic Programming and Evolvable Machines, Vol. 4. 2003. pp. 21-51. Kluwer Academic Publishers.

[6]. F. Fernandez, G. Galeano, J. A. Gómez, J. L. Guisado, "Control of bloat in Genetic Programming by means of the Island Model", VIII Parallel Problem Solving from Nature Conference. LNCS 3242. pp. 263-271. Springer

[7]. M. Tomassini, L. Vanneschi, L. Bucher and F. Fernandez "An MPI-Based Tool for Distributed Genetic Programming", Proceedings of the IEEE International Conference on Cluster Computing (Cluster2000), 209-216, IEEE Computer Society, Piscataway, NJ, 2000.

[8]. M.E. Samples, J. M. Daida, M. Byom, M. Pizzimenti, « Parameter Sweeps for Exploring GP parameters ». In Proceedings GECCO 2005.

[9]. D.P. Anderson, "BOINC: A system for Public-Resource Computing and Storage", 5th IEEE/ACM International Workshop on Grid Computing, pp. 365-372, Nov. 8 2004, Pittsburgh, PA.

[10]. I. Foster, C. Kesselman (eds) The Grid 2. Morgan Kaufmann 2005.

[11]. J. Koza, Genetic Programming: On the Programming of Computers by means of Natural Selection. Mit Press. 1992.

[12]. D. Zongker, B. Punch, and B. Rand. LilGP 1.01. user manual. Technical report, Michigan State University, 1996.

[13]. S. Cahou, E. Talbi, M. Melab. "ParadisEO: A framework for Paralled and Distributed Metaheuritics". International Paralled and Distributed Processing Symposium. (IPDPS'03).