Mecánica Computacional Vol. XXIII
G.Buscaglia, E.Dari, O.Zamonsky (Eds.)
Bariloche, Argentina, November 2004

PDAGMO PARA CONFIGURACIÓN INICIAL DE SENSORES EN PROCESOS INDUSTRIALES

Fernando Asteasuain[*], Jessica A. Carballido[*†], Gustavo E. Vazquez[*†], Ignacio Ponzoni[*†] y Nélida B. Brignole[*†]

[*]Grupo de Investigación y Desarrollo en Computación Científica (GIDeCC)
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur, Av. Alem 1253, 8000, Bahía Blanca, Argentina.

[†]Planta Piloto de Ingeniería Química - CONICET
Complejo CRIBABB, Camino La Carrindanga km.7 CC 717, Bahía Blanca, Argentina.
fa@cs.uns.edu.ar; jac@cs.uns.edu.ar; gvazquez@criba.edu.ar;
ip@cs.uns.edu.ar; dybrigno@criba.edu.ar

Palabras claves: Ingeniería de Procesos, Procesamiento Paralelo, Algoritmos Genéticos Multiobjetivo, Instrumentación, PVM.

Resumen. *En este trabajo se presenta una implementación paralelo-distribuida de un algoritmo genético multiobjetivo (pdAGMO), desarrollado para efectuar la selección de la configuración inicial de sensores en el diseño de instrumentación de plantas de procesos. El pdAGMO fue diseñado empleando el modelo evolutivo de islas y el paradigma master-worker, mientras que para su implementación se empleó la librería de pasaje de mensajes PVM (Parallel Virtual Machine). El desempeño del pdAGMO fue evaluado a través de su aplicación a un caso de estudio industrial correspondiente a una planta de producción de amoníaco. Los resultados alcanzados son muy satisfactorios en términos de speed-up, eficiencia y calidad del diseño de instrumentación.*

1 INTRODUCCIÓN

Las plantas industriales están físicamente conformadas por equipos, tales como reactores, bombas, columnas de destilación y otros, conectados a través de corrientes. El Diseño de Redes de Sensores (DRS) consiste en determinar la cantidad, tipo y ubicación de instrumentos de medición a colocar en el proceso, a fin de obtener un buen conocimiento de su funcionamiento[1].

A partir de la topología de la planta se puede construir un modelo matemático, integrado por ecuaciones algebraicas no lineales, que represente su funcionamiento en estado estacionario. El análisis de las variables que conforman el modelo constituye uno de los enfoques más empleados para encarar el DRS. En estos métodos, el estudio comienza con la definición de una configuración inicial de instrumentos establecida por un ingeniero. Sobre la base de este primer diseño, el conjunto de variables que contiene el modelo puede ser particionado en dos categorías:

a) *variables medidas*, integrada por todas las variables cuyo valor se conocerá directamente a través de los sensores incluidos en la configuración inicial de instrumentación, y
b) *variables no medidas*, subconjunto formado por las restantes variables del modelo.

Una vez definida esta clasificación, es necesario saber cuáles de las variables no medidas serán observables (es decir, cuáles podrán ser calculadas a través de las mediciones y ecuaciones del modelo). Esto permitirá establecer el grado de conocimiento que se tendrá del proceso con la configuración de instrumentos definida inicialmente. A este tipo de estudio se lo conoce como análisis de observabilidad (AO). Según los resultados alcanzados por el AO, el ingeniero podrá decidir modificar la configuración de instrumentación propuesta inicialmente, a fin de volver observables aquellas variables no medidas que no pueden ser calculadas con los sensores ya ubicados. Esto hace que la etapa del AO se vuelva un proceso iterativo, tal como se esquematiza en la Figura 1.

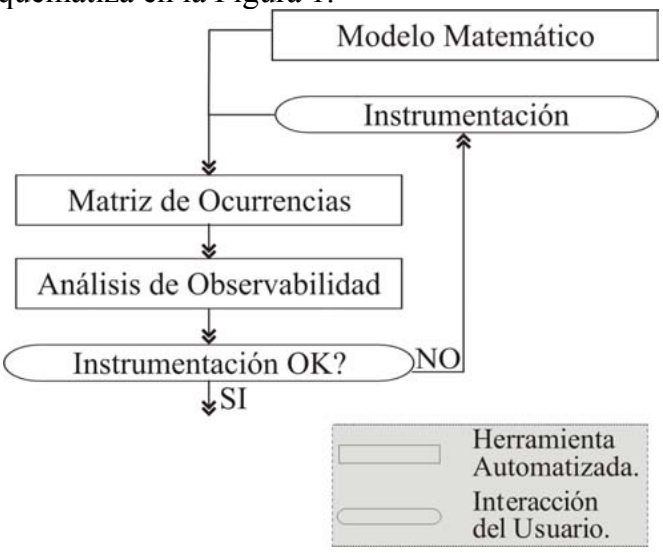

Figura 1: Etapa de Análisis de Observabilidad sin inicialización automática.

De los estudios presentados por Ponzoni et al.[2], se concluye que los algoritmos estructurales proveen la mejor alternativa para realizar el AO. Esto se debe a que permiten una mayor independencia del grado de no linealidad del modelo matemático que representa el comportamiento en estado estacionario de la planta.

Sin embargo, varios de los métodos estructurales se basan en técnicas de exploración combinatorial, las cuales implican tiempos de cómputo considerables. Si a esto le sumamos la naturaleza iterativa de toda la etapa de observabilidad, se vuelve imprescindible contar con una inicialización de instrumentos que tienda a reducir la cantidad de veces que se ejecuta el algoritmo para AO. En la figura 2 se puede ver como se realizaría el AO si utilizamos una herramienta automática para obtener la configuración inicial.

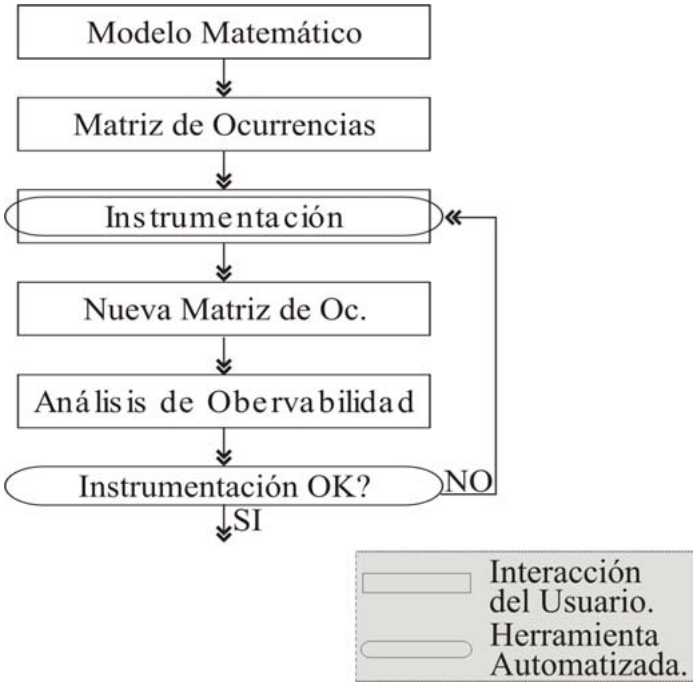

Figura 2: Etapa de Análisis de Observabilidad con inicialización automática.

El problema de encontrar una buena inicialización persigue varios objetivos, algunos contrapuestos, tales como la maximización de confiabilidad y observabilidad de la configuración, así como la minimización de su costo. A su vez el método utilizado para obtener dicha configuración debe ser rápido en cuanto a tiempos de cómputo por tratarse de una técnica de inicialización. Por todas estas razones resultó natural pensar en la utilización de un algoritmo genético multi-objetivo (AGMO) como herramienta de inicialización, ya que los mismos se caracterizan por obtener buenos resultados con tiempos de ejecución relativamente bajos [3].

2 ALGORITMO GENÉTICO MULTI-OBJETIVO

Los datos de entrada requeridos por el AGMO son la matriz de ocurrencia (de MxN, donde M es la cantidad de ecuaciones y N es la cantidad de variables) obtenida a partir del modelo matemático de la planta, dos vectores -vc y vr- que contienen los costos y confiabilidad correspondientes a los sensores necesarios para medir cada una de las variables en ese modelo, y un vector binario de penalizaciones, donde un 1 significa que esa variable no debería ser medida (por ejemplo si se trata de una entalpía). Cada individuo del AGMO representa una posible configuración de instrumentos mediante una cadena binaria de longitud N donde un 1 implica una medición. Los operadores de cruzamiento y mutación son los clásicos[4]. La función de aptitud se implementó con un enfoque non-pareto de agregación, combinando todos los objetivos en un solo valor a maximizar[5]. La forma final de la fórmula de la función de aptitud permite definir dentro de un rango entre 0 y *n* a los *n* objetivos tenidos en cuenta. En este caso particular, en que se consideraron costo, confiabilidad y observabilidad, los valores resultantes de la función de aptitud están siempre en el rango [0..3]. Para obtener mayor detalle sobre la versión secuencial del AGMO puede consultarse Carballido et al.[6].

3 ALGORITMOS GENÉTICOS PARALELOS

Los algoritmos genéticos poseen un alto grado de paralelismo implícito. Esto se debe a que la mayoría de las operaciones de variación pueden desarrollarse en forma simultánea, haciendo que su extensión paralela resulte muy natural y extremadamente simple. Además de la ventaja de una superior eficiencia en tiempos, los modelos paralelos permiten un mejor desempeño numérico debido al uso de una población distribuida[7]. En particular, en el trabajo de Van Veldehuizen et al.[8], se analizan las bondades de los algoritmos genéticos paralelos en el tratamiento de problemas multiobjetivo, mientras que en el trabajo de Sena et al.[9] se presenta un interesante ejemplo de algoritmo genético paralelo implementado mediante la librería de pasaje de mensajes PVM[10].

Dentro de los modelos paralelos empleados en Computación Evolutiva, existe una clasificación según la manera en que el paralelismo se lleva a cabo y la forma en que se distribuye su población. Las formas más conocidas de algoritmos genéticos paralelos son el modelo de islas y el modelo celular[11]. En el primero, la población es distribuida en "islas" y la evolución de cada subpoblación se lleva a cabo concurrentemente, con eventuales migraciones periódicas de individuos entre las distintas islas. Generalmente, en cada isla la población posee varios individuos y su implementación es usualmente realizada sobre arquitecturas MIMD (Multiple Instruction-Multiple Data). En cambio, en el modelo celular, cada subpoblación consta de un individuo ubicado en una celda, la cual forma parte de una grilla que interconecta varias celdas. Este enfoque es implementado de manera natural sobre plataformas SIMD (Single Instruction-Multiple Data). En ambos enfoques nuevos parámetros entran en juego: la migración o intercambio de individuos de una subpoblación a otra, cantidad y frecuencia de la misma, la selección y método de reemplazo de los individuos migrados, y la topología que conecta a las subpoblaciones.

3.1 Implementación del pdAGMO

Ambiente de desarrollo. En función de la infraestructura disponible, se decidió implementar nuestro algoritmo paralelo genético multi-objetivo siguiendo el modelo de islas, y el paradigma *master-worker* sincrónico como modelo de comunicación para el procesamiento distribuido.

Básicamente, la población es particionada en varias subpoblaciones. La utilización del paradigma *master-worker* establece que cada subpoblación reside en una isla cuya evolución es computada por un *worker*, mientras que el *master* es el encargado de controlar y sincronizar el cómputo de los *workers*. Las políticas de migración determinan el intercambio de individuos entre las islas, lo que les permite compartir material genético.

La estrategia de comunicación es implementada a través de la librería Parallel Virtual Machine (PVM) sobre una red de computadoras de área local. Se optó por PVM debido a su aceptación a nivel mundial, y a que permite manejar recursos computacionales heterogéneos como una única máquina paralela, incluyendo controles de fallas e interoperabilidad.

pdAGMO: Estructura General. La idea general del pdAGMO sigue un esquema *master-worker* clásico. La principal función del *master* es coordinar la tarea de los *workers*. Crea los *workers* que sean necesarios y les indica los parámetros del problema. Cada *worker* recibe los parámetros del *master*, genera al azar su propia subpoblación, y ejecuta para la misma el AGMO. Al finalizar cada ciclo evolutivo, los *workers* envían al *master* el mejor individuo de su generación. Esta información le es suficiente al *master* como para determinar si el objetivo está satisfecho o si, por el contrario, es necesario continuar. Dentro del ciclo evolutivo, es posible que se realice una migración, es decir, que se realice un intercambio de individuos entre las islas según criterios bien definidos de migración.

Parámetros del pdAGMO. Podemos clasificar los parámetros del algoritmo de la siguiente manera, aquellos propios de la paralelización y los pertenecientes al AGMO. Dentro de los primeros surgen: el número de *workers*, el tamaño de la población, el intervalo de migración, esto es, cada cuántas generaciones realizar una migración, cuántos individuos migrar, y el error mínimo aceptado, que determina la condición de corte del algoritmo. Este último requiere tener un conocimiento aproximado de la solución óptima al problema. En la segunda categoría están los parámetros clásicos de los algoritmos genéticos, como son las condiciones para el cruzamiento y la mutación.

Funcionamiento del Master. El funcionamiento del *master* puede ser descripto por el siguiente seudo-código:

```
Crear_Workers();
Enviar Parámetros_del_Problema_a_Workers();
Mientras (Error > Error_Minimo) y (generacion < GenMaxima) Hacer
      Recibir_mejores_de_cada_worker();
      Computar_error_actual();
Fin Mientras
Detener_Workers();
Mostrar_Mejor_Individuo();
```

Lo primero que hace el *master* es crear los *workers* y distribuirles a todos los parámetros del problema. Luego entra en un ciclo hasta arribar a una solución o bien, hasta exceder el número de generaciones permitido. En cada ciclo recibe el mejor individuo de cada *worker*, obtiene el mejor de ellos, y lo compara contra una posible solución óptima. Si el error obtenido satisface el error mínimo, entonces termina y se encarga de finalizar la tarea de los *workers*, sino, itera una vez más. Esta condición de corte es fundamental, porque permite una comparación justa entre la versión secuencial y la paralela al momento de cuantificar el *speed-up*. El fin es asegurar que ambas versiones encuentren soluciones de similar calidad. Esta forma de establecer la condición de corte es empleada comúnmente en algoritmos genéticos paralelos[12,13].

Funcionamiento del Worker. El funcionamiento del *worker* puede ser descripto por el siguiente seudo-código:

```
Recibir_Parametros_del_Problema();
Generar_Población();
Mientras (Verdadero) Hacer
      Computar_Algoritmo_Genetico;
      Si (es requerido) Realizar_Migracion();
      Enviar_Mejor_al_Master();
Fin Mientras
```

Cada *worker* recibe los parámetros del problema y crea al azar su propia población. Luego, entra en un ciclo infinito, muy similar al ciclo de un AGMO. La diferencia está en que, al finalizar, es posible realizar una migración y además el mejor individuo es enviado al *master*. Por razones de eficiencia este ciclo es infinito, y cada *worker* se detendrá únicamente cuando el *master* los finalice una vez que se satisfaga la condición de terminación.

La política de migración es la propuesta en Sena et al [9], que consiste en mover el material genético entre las islas como si los *workers* estuvieran conectados en una topología de anillos. Cuando llega el momento de migrar, cada *worker* envía su mejor individuo al *worker* siguiente (como es topología de anillos, el último *worker* envía al primer *worker*). Luego, cada *worker* recibe un individuo del *worker* anterior, y reemplaza con el mismo al peor individuo de su generación. Luego de varias migraciones cada *worker* tendrá los mejores individuos encontrados por los demás *workers*.

4 CASO DE ESTUDIO: UNA PLANTA DE SÍNTESIS DE AMONÍACO

La planta considerada como caso de estudio fue diseñada por Bike[14] con la finalidad de producir 1500 ton/día de amoníaco líquido anhidro a 240 K y 450 kPa, con una pureza mínima de 99.5%. El proceso utilizado es el de Haber-Bosch (síntesis a media presión), mediante el cual el amoníaco es recuperado por absorción con agua. La planta incluye también una sección de recuperación de hidrógeno, el cual una vez recuperado es mezclado con la alimentación para ser utilizado nuevamente.

Las alimentaciones son corrientes de hidrógeno y nitrógeno provenientes de un proceso de gasificación de carbón, conteniendo impurezas (argón y metano). Ambas corrientes son comprimidas y mezcladas luego con la corriente proveniente de la sección de recuperación de

hidrógeno. La corriente resultante es llevada a las condiciones de operación del reactor catalítico (21000kPa, 703K) y alimentada al reactor. El producto es alimentado a un flash, cuya corriente gaseosa es alimentada al absorbedor y la líquida es mezclada con la salida del absorbedor y alimentada a la torre de destilación, donde se obtiene el producto deseado a 450kPa, 240 K y una pureza del 99.5%, como corriente de tope.

La corriente de fondo (99.9% de agua), es reciclada al absorbedor. La corriente gaseosa a la salida del absorbedor es secada y luego reciclada en un 96.4% al reactor, mientras que el resto es alimentado a la sección de recuperación de hidrógeno. En ésta sección, cada separador contiene membranas permeables, y está diseñado para recuperar un 80% de hidrógeno, el cual es luego reciclado y mezclado con las corrientes de alimentación al proceso. Los gases no reciclados son purgados y comercializados como combustible.

El modelo matemático de esta planta fue generado utilizando el software ModGen[15], obteniéndose un sistema de ecuaciones algebraicas no lineales formado por 546 variables y 557 ecuaciones. Un rasgo distintivo del algoritmo genético adoptado para analizar la planta de amoníaco es que se introdujeron consideraciones específicas, tendientes a garantizar la factibilidad de los resultados finales desde el punto de vista de su real implementación en planta. En principio, un individuo contiene absolutamente todas las variables de estado que surgen del modelo matemático seleccionado para representar el comportamiento de la planta. A pesar de esto, muchas de esas variables (tales como entalpías y constantes de equilibrio) no pueden ser medidas en forma directa. Por lo tanto, para el tratamiento de problemas industriales, resulta necesario introducir penalidades para "castigar" a los individuos que representan configuraciones físicamente no factibles. La política adoptada en este sentido fue la de eliminarlos directamente, asignándoles *fitness* cero.

5 RESULTADOS EXPERIMENTALES

El trabajo experimental se efectuó en PCs Pentium IV de 2.4 GHz y 512 MB de RAM, conectadas a través de una red FastEthernet de 100Mb/s. Se ejecutó el pdAGMO empleando 1, 2, 4, y 8 procesadores. El tamaño de población para la versión secuencial fue establecida en 128 individuos, mientras que los tamaños de las subpoblaciones en las corridas en paralelo fueron establecidas en forma proporcional en función de la cantidad de islas (procesadores), tal como se muestra en la tabla 1. El número total de corridas ejecutadas para los distintos modelos fue 100.

Número de Procesadores	2	4	8
Tamaño Subpoblación	64	32	16

Tabla 1. Tamaño de subpoblación según la cantidad de procesadores.

Para la experimentación con el pdAGMO se trabajó con islas heterogéneas, es decir, las distintas islas corrieron con valores de parámetros diferentes. En la mitad de las islas prevaleció la explotación, aplicando una mayor presión selectiva, la cual fue lograda mediante el aumento de la probabilidad de cruzamiento (75%), y una menor diversidad genética, que se alcanzó a través de la reducción en la probabilidad de mutación (10%). En cambio, en la otra

mitad de las islas, se favoreció la exploración, usando una menor probabilidad de cruzamiento (65%), y aumentando la probabilidad de mutación (15%). En todos los casos, la probabilidad de mutación disminuyó dinámicamente con el número de generaciones. Por otro lado, las migraciones se realizaron cada cinco generaciones.

Esta heterogeneidad del modelo de islas llevó a realizar las pruebas con dos versiones del algoritmo AGMO secuencial, uno con los parámetros de Exploración (Secuencial Exploración) y el restante con los parámetros de Explotación (Secuencial Explotación), para así obtener una comparación más justa.

El umbral establecido para la condición de corte fue de 2,62. Es decir, tanto los AGMO secuenciales como el pdAGMO detienen la evolución cuando encuentran un individuo con valor de *fitness* igual o superior a 2,62, o bien, al arribar al número máximo de generaciones permitido, valor que fue fijado en 300. Estos valores para la condición de corte surgieron del análisis experimental efectuado con anterioridad en Carballido et al.[6].

Las mejores configuraciones iniciales de instrumentos obtenidas por el algoritmo paralelo distribuido resultaron siempre satisfactorias en término de los criterios ingenieriles. En otras palabras, todas las variables relevantes del proceso industrial siempre quedaron clasificadas dentro de los conjuntos de variables medidas u observables a partir de la instrumentación propuesta por el pdAGMO.

En la tabla 2 se muestran los resultados obtenidos en cuanto a la calidad de la solución hallada por cada modelo (medido en valor de *fitness*). Los números mostrados en las entradas de la tabla indican el porcentaje de corridas en que el *fitness* del mejor individuo fue mayor o igual a un determinado umbral. Solo se consideraron los umbrales de 2,58 a 2,62.

Umbrales	Secuencial Exploración	Secuencial Explotación	2 *workers* Población: 64	4 *workers* Población: 32	8 *workers* Población: 16
%veces > 2,62	4	0	5	24	71
%veces > 2,61	8	0	11	36	85
%veces > 2,60	12	12	30	63	93
%veces > 2,59	24	28	55	83	96
%veces > 2,58	42	54	73	89	99

Tabla 2. Comparación de la calidad de la solución encontrada.

En esta tabla se ve reflejado cómo los modelos paralelos encuentran soluciones de alta calidad en un porcentaje mucho mayor que los modelos secuenciales. A su vez, dentro de los modelos paralelos, los porcentajes son mayores a medida que el número de islas aumenta y el número de la población disminuye. Esto se debe a que al aumentar la cantidad de islas, se incrementa la presión selectiva, dado que los mejores individuos de cada isla se replican un mayor número de veces. Esto acelera la convergencia del algoritmo genético a soluciones de mejor *fitness*. Los resultados de la tabla anterior se observan gráficamente en la figura 3.

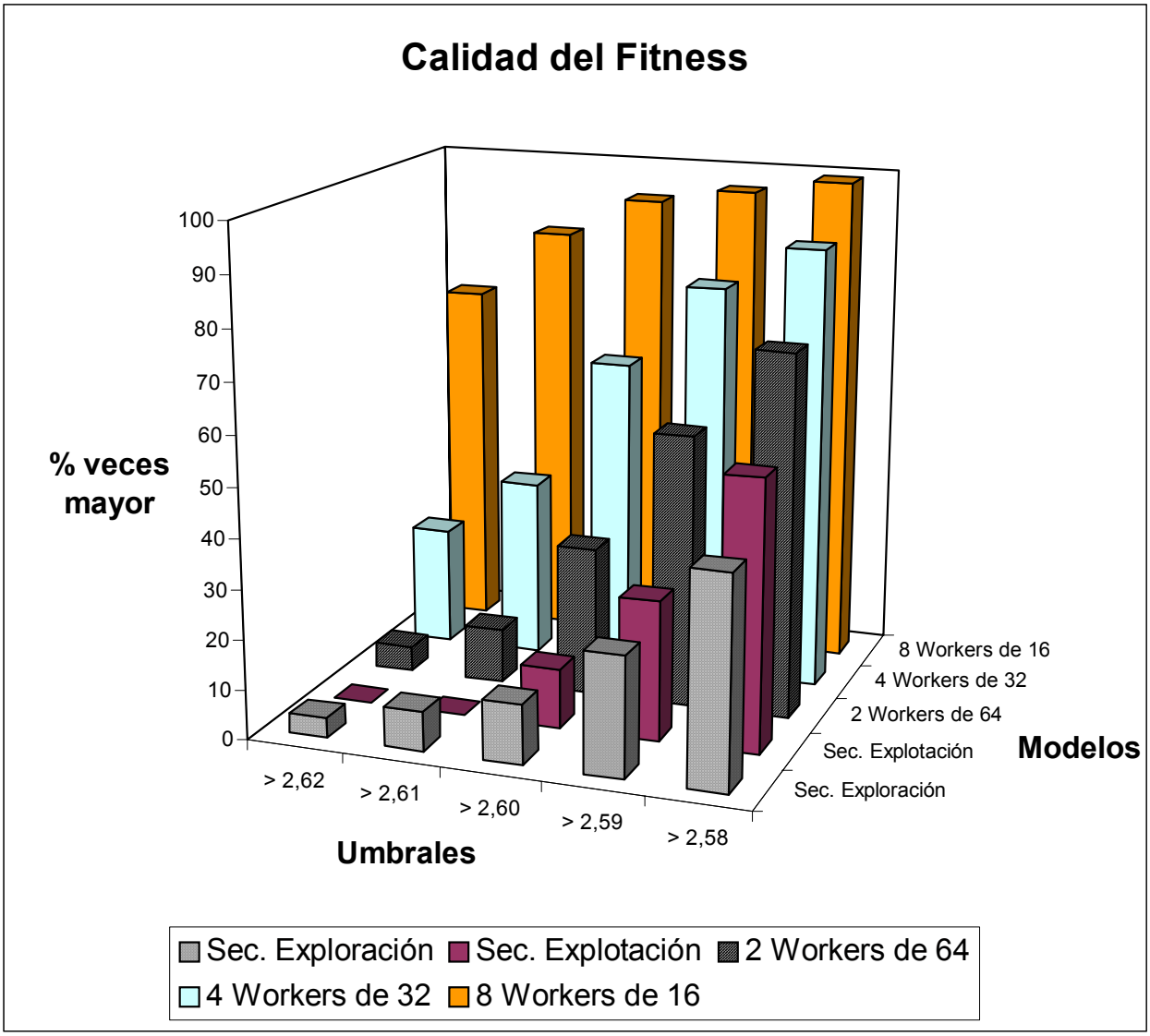

Figura 3. Gráfico comparativo de la calidad de la solución encontrada por los distintos modelos.

La tabla 3 muestra el tiempo de cómputo promedio, el mejor *fitness* promedio y la generación de convergencia promedio de las corridas efectuadas en este trabajo experimental para cada modelo. Observando los datos de la tabla 3, se aprecia una importante reducción en el tiempo requerido para obtener el mejor individuo, así como también en la cantidad de generaciones necesarias para encontrarlo, a medida que crece la cantidad de procesadores empleados por el pdAGMO. Sumado a esto, se agrega el hecho que el *fitness* promedio aumenta al incrementar la cantidad de islas. Nuevamente, aumentar la cantidad de islas posee un impacto no solo en la calidad del *fitness* promedio sino también en la disminución de la generación de convergencia promedio debido a la mayor presión selectiva. Las figuras 4, 5 y 6 reflejan de manera gráfica los tiempos de ejecución promedio, la generación de convergencia promedio, y el mejor *fitness* promedio respectivamente.

Modelo	*Fitness* Promedio	Gen. Conv. Promedio	Tiempo Promedio (seg)
Sec. Exploración	2,576381	265,94	175,2
Sec. Explotación	2,579004	268,48	175,2
2 *workers* de 64	2,589928	243,01	80,7
4 *workers* de 32	2,602757	201,97	38,7
8 *workers* de 16	2,617313	155,08	19,4

Tabla 3. Comparación de *fitness*, generación de convergencia y tiempo de cómputo.

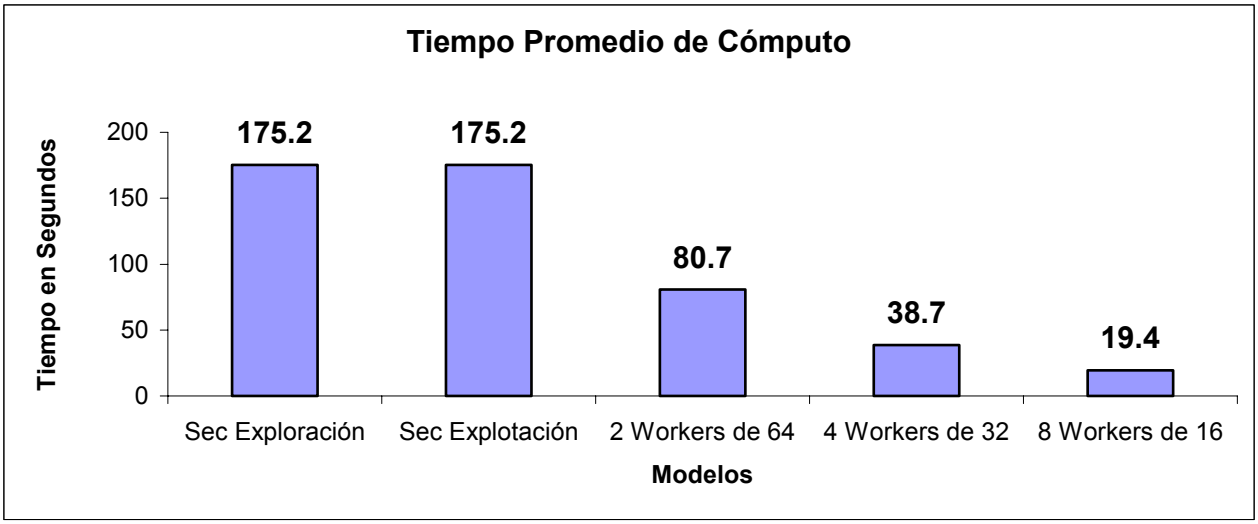

Figura 4. Tiempos de Ejecución Promedio.

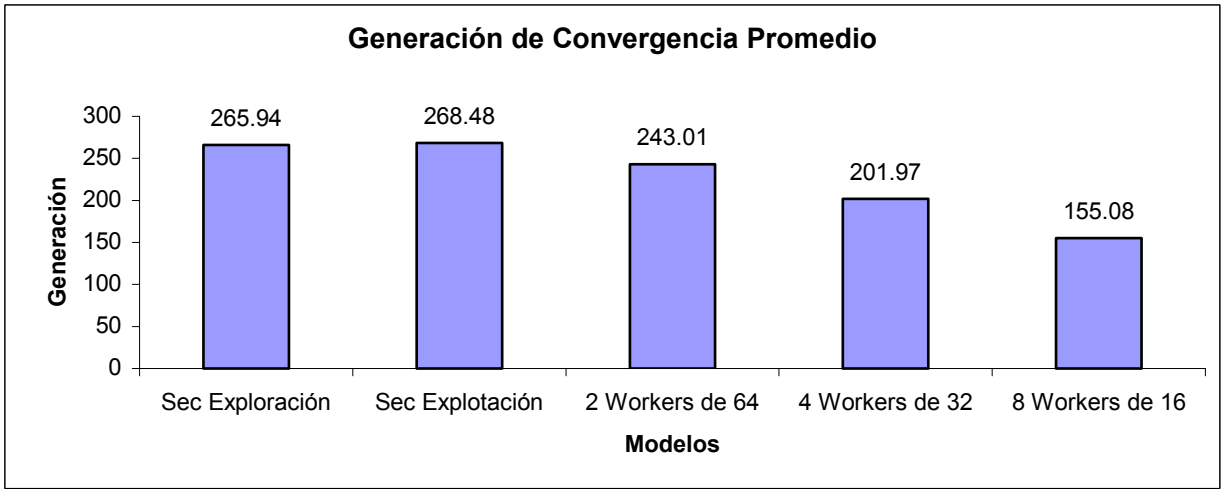

Figura 5. Generación de Convergencia Promedio.

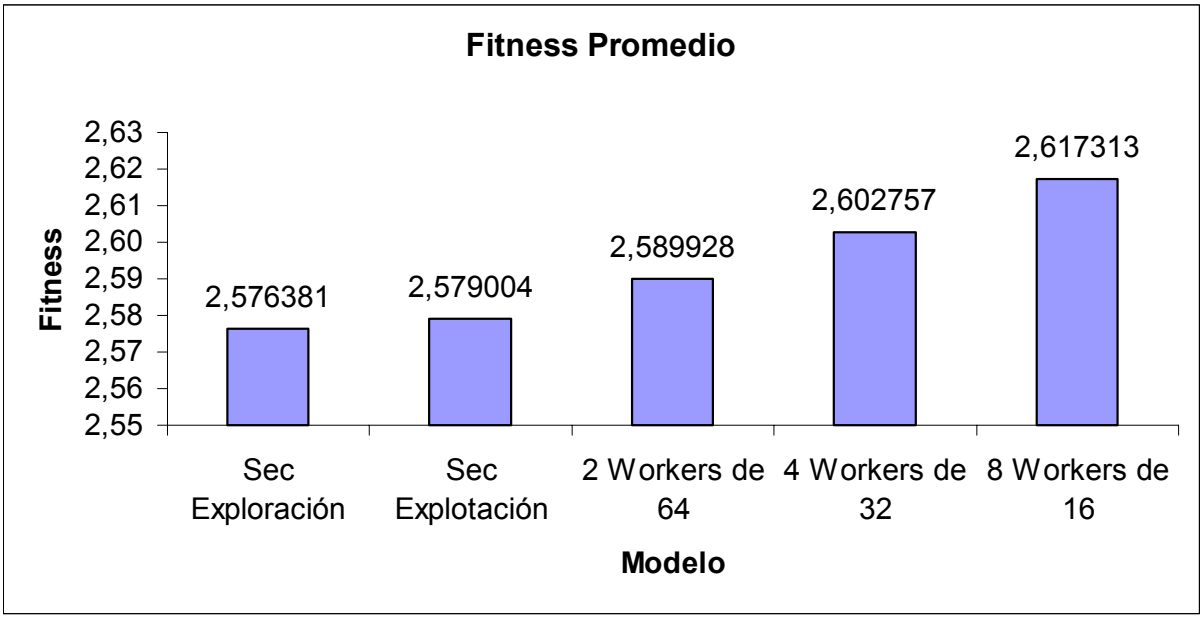

Figura 6. Mejor *Fitness* Promedio.

Los datos de la tabla 4 muestran los excelentes resultados de *speed-up* y eficiencia logrados por el pdAGMO. La escalabilidad del sistema paralelo es muy buena. Tal como se observa en la segunda columna de la tabla 4, en todos los casos se logra un *speed-up* súper lineal. Si bien a priori este resultado puede parecer controversial, ya ha sido establecido que los algoritmos genéticos paralelos basados en modelos de islas pueden alcanzar, tanto teóricamente como en la práctica, *speed-up* súper lineales[12]. En este caso en particular, el *speed-up* súper lineal se alcanza debido a que la descomposición de la población en subpoblaciones más chicas permite una ganancia que es de orden superior a lineal[7].

Modelo	*Speed-up*	Eficiencia
2 *workers* de 64	2,17	108%
4 *workers* de 32	4,58	113%
8 *workers* de 16	9,03	112%

Tabla 4. Resultados de *speed-up* y Eficiencia.

6 CONCLUSIONES Y TRABAJO FUTURO

En este trabajo se presentó la implementación paralelo distribuida de un algoritmo genético multiobjetivo (pdAGMO) diseñado para realizar la configuración inicial de sensores en el diseño de instrumentación de procesos industriales.

El pdAGMO fue diseñado siguiendo el modelo evolutivo de islas heterogéneas, donde en cada isla se representa la evolución de una subpoblación. La implementación utiliza la librería de pasaje de mensajes PVM, mientras que la arquitectura del programa sigue un esquema *master-worker*.

El algoritmo fue testeado en el diseño de la configuración inicial de instrumentos de una

planta de producción de amoníaco. Los resultados arrojados por el pdAGMO fueron excelentes, tanto por la calidad de la instrumentación obtenida, como por los valores de *speed-up* y eficiencia alcanzados.

Si bien los resultados son muy alentadores, hay varios aspectos de la implementación y evaluación del pdAGMO que deben ser analizados a futuro. El primero de ellos es el impacto de variar el intervalo de migración, dado que este parámetro posee una fuerte incidencia en la presión selectiva del algoritmo. Otro aspecto vinculado con la migración es la modificación del protocolo de comunicación sincrónica actual hacia un esquema asincrónico, a fin de aumentar el nivel de concurrencia del algoritmo. Por otra parte se proyecta incrementar el nivel de heterogeneidad de las islas utilizando parámetros adaptivos basándonos en las propuestas efectuadas por Eiben et al.[16] y Tongchim y Chongstitvatana[17]. Finalmente se planifica evaluar el desempeño del pdAGMO en problemas industriales de mayor escala.

7 AGRADECIMIENTOS

Los autores desean expresar su agradecimiento a la Agencia Nacional de Promoción Científica y Tecnológica de la Argentina por la subvención otorgada en el marco del Programa de Modernización Tecnológica, Contrato de Préstamo BID 1201/OC-AR, al Proyecto de Investigación Científica y Tecnológica Nº11-12778, denominado "Procesamiento paralelo distribuido aplicado a ingeniería de procesos", aprobado por Resolución ANPCYT Nº117/2003.

También queremos agradecer a la Secretaría de Ciencia y Tecnología de la Universidad Nacional del Sur por la subvención otorgada al Proyecto de Grupos de Investigación (PGI 24/N012): "Computación Científica Aplicada al Diseño de Instrumentación".

8 REFERENCIAS

[1] J.A. Romagnoli and M.C. Sánchez, "Data Processing and Reconciliation for Chemical Process Operations",Academic Press,San Diego, USA (1999).

[2] I. Ponzoni, M.C. Sánchez and N.B. Brignole, "A New Structural Algorithm for Observability Classification", *Ind. Eng. Chem. Res,* **38***,* 3027-3035 (1999).

[3] D.E. Goldberg, *"Genetic Algorithm in Search, Optimization, and Machine Learning"*, Addison-Wesley (1999).

[4] Z. Michalewicz, *"Genetic Algorithms + Data Structures = Evolutions Programs"*, Springer-Verlag (1996).

[5] C.A. Coello Coello, D.A. Van Veldhuizenand and G.B. Lamont, *"Evolutionary Algorithms for Solving Multi-Objective Problems"*, Kluwer Academic Publishers (2002).

[6] J.A. Carballido, I. Ponzoni. and N.B. Brignole, "Un Algoritmo Genético Multi-Objetivo para Diseño Inicial de Redes de Sensores", *ENIEF 2003,* B.Blanca (Argentina) del 4-7 nov. 2003. Publicado en *Mecánica Computacional*, **22**, 1295-1304, AMCA (2003).

[7] E. Alba and M. Tomassini, "Parallelism and Evolutionary Algorithms", *IEEE Trans. on Evolutionary Comput*, **6**, 443-462, (2002).

[8] D.A. Van Veldhuizen, J.B. Zydallis and G.B. Lamont, "Considerations in Engineering Parallel Multiobjective Evolutionary Algorithms", *IEEE Trans. on Evolutionary Comput.*, **7**, 144-173, (2003).

[9] G.A. Sena, D. Megherbi and G. Isern, "Implementation of a parallel Genetic Algorithm on a cluster of workstations: Traveling Salesman Problem, a case study", *Future Generation Computer Systems*, **17**, 477-488, (2001).

[10] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Mancheck and V. Sunderam, *"PVM: Parallel Virtual Machine. A users guide and tutorial for network parallel computing"*, MIT Press (1994).

[11] J. Stender, *"Parallel Genetic Algorithms: Theory & Applications"*, Capítulo 1, IOS Press, Amsterdam, Holanda (1993).

[12] E. Alba, "Parallel evolutionary algorithms can achieve super-linear performance", *Information Processing Letters*, **82**, 7-13, (2002).

[13] E. Cantú-Paz and Goldberg D.E., "Predicting speedups of idealized bounding cases of parallel genetic algorithms", *7 International Conference on Genetic Algorithms,* Morgan Kaufmann, Los Altos, CA,338-345,(1997).

[14] S. Bike., *"Design of an Ammonia Synthesis Plant, CACHE Case Study"*, Department of Chemical Engineering, Carnegie Mellon University, (1985).

[15] G.E. Vazquez, I. Ponzoni, M.C. Sánchez and N.B. Brignole, "ModGen: A Model Generator for Instrumentation Analysis", A*dvances in Engineering Software,* **32**,37-48, (2001).

[16] Á.E. Eiben, R. Hinterding and Z. Michalewicz, "Parameter control in evolutionary algorithms", *IEEE Trans. Evolutionary Comput*, **3**, 124-141 (1999).

[17] S. Tongchim and P. Chongstitvatana, "Parallel genetic algorithm with parameter adaptation", *Information Processing Letters*, **82**,47-54,(2002).