UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

Titulo: Extensión del entorno CMRE mediante la implementación de conceptos relacionados a concurrencia y paralelismo. Su utilización en el ámbito educativo.
Autores: Castro, Juan Manuel
Director: Dra. Laura De Giusti
Codirector: Esp. Gladys Gorga
Carrera: Licenciatura en Informática

Resumen

Se presenta el desarrollo de una ampliación funcional al entorno CMRE (Concurrent Multi Robot Environment) denominada ECMRE (Extended Concurrent Multi Robot Environment). La herramienta educativa ECMRE incorpora algunos conceptos relacionados a los ya existentes de Concurrencia y Paralelismo para su utilización en asignaturas de los primeros años de Informática. El alumno puede ejecutar un algoritmo con robots que trabajan a distinta velocidad, e incorporan tiempo de ejecución, consumo energético y temperatura. Se permite acelerar/disminuir la velocidad de los robots (potencia de cómputo) en base al estado (consumo energético y temperatura) de los mismos durante la ejecución.

Se exponen los primeros resultados obtenidos a partir de la experiencia realizada con alumnos de primer año en la materia Taller de Programación de la Facultad de Informática de la UNLP. Se presentaron los contenidos teóricos incorporados al entorno y se trabajó con la herramienta ECMRE, mediante la implementación de un ejercicio práctico. Por último, se realizó una encuesta a los alumnos obteniendo así un primer feedback de la aplicación

Palabras Claves

- *Concurrencia*
- *Paralelismo*
- *Procesadores heterogéneos*
- *Algoritmos paralelos*
- *Consumo energético*
- *Temperatura*
- *CMRE*
- *ECMRE*

Trabajos Realizados

- *Investigación de conceptos de procesadores multicore y sus arquitecturas, como el consumo de energía, temperatura de los procesadores, balance de carga de la aplicación, entre otros.*
- *Definición e implementación del prototipo ECMRE que incorpora nuevos conceptos de Concurrencia y Paralelismo al entorno CMRE.*
- *Presentación del entorno ECMRE en la materia Taller de Programación de la Facultad de Informática de la UNLP.*

Conclusiones

- *Se desarrolló una ampliación funcional al entorno CMRE denominada ECMRE, la cual permite ejecutar un algoritmo con robots que trabajan a distinta velocidad, e incorpora tiempo de ejecución, consumo energético y temperatura por cada procesador/robot, entre otros cambios adicionales.*
- *Se obtuvo una evaluación positiva de la herramienta por parte de los alumnos de Taller de Programación, a los que se presentó el entorno y realizo una encuesta.*

Trabajos Futuros

- *Continuar la evaluación de la herramienta, publicando la aplicación ECMRE para docentes y alumnos. Proveer manual de usuario y una vía de comunicación para canalizar y atender posibles consultas en cuanto a la utilización del entorno, reportes de bugs, mejoras funcionales, entre otros.*
- *Evaluar nuevos desarrollos como la incorporación de otras características al robot para representar con mayor detalle los procesadores actuales, ampliar el módulo Log del Procesador, entre otros.*

Fecha de la presentación: SEPTIEMBRE 2017

UNIVERSIDAD NACIONAL DE LA PLATA
FACULTAD DE INFORMÁTICA

EXTENSIÓN DEL ENTORNO CMRE MEDIANTE LA IMPLEMENTACIÓN DE CONCEPTOS RELACIONADOS A CONCURRENCIA Y PARALELISMO. SU UTILIZACIÓN EN EL ÁMBITO EDUCATIVO.

Alumno: Juan Manuel Castro
Director: Dra. Laura De Giusti
Codirector: Esp. Gladys Gorga

CAPITULO 1
INTRODUCCION

1.1 Prefacio

La presente tesina realiza una ampliación funcional y conceptual al entorno CMRE (Concurrent Multi Robot Environment), que se utiliza en diversas cátedras de carreras de Informática de universidades nacionales, para abordar la enseñanza y el aprendizaje de conceptos básicos de programación en los primeros años de la carrera, y conceptos complejos de concurrencia en años más avanzados. Esta extensión es llamada ECMRE (Extended Concurrent Multi Robot Environment).

Para mantener una currícula actualizada, las cátedras plantean la necesidad de incorporar estos nuevos conceptos basados en arquitecturas de hardware paralelas, que resultarán de utilidad a los alumnos durante el transcurso de la carrera.

Se describen y analizan los principales aspectos que emergen a partir del surgimiento de los procesadores con múltiples núcleos. Se realiza una introducción a los conceptos de concurrencia y paralelismo, y se plantea como ha afectado el cambio tecnológico en la enseñanza de la programación poniendo el foco principalmente en aquellas problemáticas vigentes en la actualidad, relacionadas con el consumo, la temperatura y la eficiencia energética necesarios para la ejecución de procesos concurrentes y paralelos.

El entorno CMRE es una herramienta interactiva, lo que resulta adecuado para la enseñanza de esos conceptos en las etapas iniciales de la carrera, dado que los mismos poseen cierto grado de complejidad en el aprendizaje.

Se detalla el entorno CMRE en su versión actual, y se presenta el diseño e implementación de un nuevo módulo que incorpora los conceptos de concurrencia y paralelismo mencionados previamente.

Por último, se presentan los resultados obtenidos a partir de las experiencias realizadas en las asignaturas de los primeros años de las carreras de la Facultad de Informática de la UNLP.

1.2 Objetivos

Objetivos Generales

- Incorporar al entorno CMRE algunos conceptos relacionados a los ya existentes de Concurrencia y Paralelismo para su utilización en asignaturas de los primeros años de Informática.

Objetivos Específicos

- Investigar conceptos de procesadores multicore y sus arquitecturas, como el consumo de energía, temperatura de los procesadores, balance de carga de la aplicación, entre otros.
- Extender el entorno CMRE (Concurrent Multi Robot Environment) hacia ECMRE (Extended Concurrent Multi Robot Environment), permitiendo la posibilidad de variar las velocidades de los múltiples robots. Esto se verá reflejado gráficamente a través del consumo y la temperatura de los mismos. Además, se permitirá acelerar/disminuir la velocidad de los robots (potencia de cómputo) en base al estado (consumo energético y temperatura) de los mismos durante la ejecución.
- Presentar los conceptos incorporados al entorno ECMRE en las asignaturas de los primeros años de las carreras de la Facultad de Informática de la UNLP. Generar un informe en base a los resultados obtenidos a partir de la utilización del entorno y mediante la realización de encuestas a los alumnos de la cátedra de Taller de Programación de la Facultad de Informática.

1.3 Motivación

El modelo de Von Neumann (Von Neumann, 1945; Burks *et al.*, 1946) fue durante mucho tiempo una referencia ideal para iniciar al alumno en el estudio de la programación. El modelo es sencillo y consistente, e incluso se mantenía vigente hasta hace algunos años atrás. Las arquitecturas multicore inicialmente fueron desarrolladas por empresas como Intel y AMD y sus procesadores se encontraban en su etapa de diseño o bien eran aplicados a supercomputadores de uso específico (Intel, 2012). Hacer una mención a este tipo de nuevas arquitecturas era suficiente para el alumno, planteando una posible evolución al modelo de Von Neumann.

En la actualidad, la evolución hacia arquitecturas multicore es un hecho (no sólo a nivel de hardware sino también de software). Las arquitecturas de procesadores, conformadas por múltiples "cores" o núcleos en un procesador físico, han sido incorporadas a productos de uso

cotidiano (Held *et al.*, 2006). En la actualidad computadoras, tablets, celulares, consolas de videojuegos entre otros, cuentan con un diseño multicore para incrementar la potencia de cálculo (Grama *et al.*, 2003; Chapman, 2007). La utilización de aplicaciones multithread es tan habitual que suele ser transparente al usuario. Por ejemplo, al escribir este documento, el procesador de texto ha utilizado multitheading ejecutando un corrector ortográfico que subraya y sugiere correcciones a medida que se edita el documento.

Se puede concluir que la "máquina de Von Neumann" con un solo hilo de control, ha perdido vigencia. Los procesadores de un sólo núcleo han sido suplantados por una nueva generación con capacidad de procesamiento paralelo (McCool, 2008). Esta generación de procesadores surgida, en las últimas décadas, ha permitido encontrar la solución a problemas de la vida real que se encontraban estancados principalmente por una limitación tecnológica. Los multicores, han permitido disminuir los tiempos de cómputo de muchas aplicaciones.

Esta evolución afectó directamente a la Informática, pero esencialmente al desarrollo de algoritmos. El principal aspecto a mencionar es la posibilidad de utilizar "concurrencia real", en donde múltiples procesadores trabajan al mismo tiempo sobre varios hilos de control (Gepner & Kowalik, 2006; McCool 2008). Se debe señalar que hasta la llegada de las arquitecturas multicore un algoritmo podía emplear de forma parcial la concurrencia, incluso al utilizar hardware muy específico como los coprocesadores. Por otro lado, el concepto de paralelismo sólo podía expresarse en un esquema de sistemas distribuidos.

Disponer de una ejecución realmente paralela a nivel físico incrementa la capacidad de procesamiento en el tiempo. Esta aceleración en el cómputo y la disminución de los tiempos han sido el eje y objetivo principal de esta evolución. Un reflejo de lo mencionado es el listado del TOP500 que utiliza la métrica FLOPS (FLoating-point OPerations per Second, cantidad de operaciones de coma flotante por segundo) para determinar el orden de clasificación de las supercomputadoras (TOP500, 2016). Sin embargo, han surgido nuevos conceptos a evaluar. Entre los más importantes se encuentra el incremento en el consumo de energía y el calor generado, ambos consecuencia directa de la aceleración del cómputo (Wu-Chun Feng, 2005). En 2007 se creó la entidad Green500, complemento del TOP500, la cual clasifica a las supercomputadoras de mayor eficiencia energética del mundo, considerando el rendimiento por W (FLOPS por Watt) al ejecutar un cierto benchmark (Green500, 2016). El consumo energético y el incremento de la temperatura son factores a tener en cuenta no sólo en el diseño del hardware sino también al desarrollar las aplicaciones.

Otro aspecto importante para analizar es la existencia de dos tipos principales de arquitecturas que afectan sustancialmente al desarrollo. Estas son las arquitecturas homogéneas, donde todos los cores poseen las mismas características, y las arquitecturas heterogéneas en las que se

pueden encontrar cores con diferentes características (Bower *et al.*, 2008). El segundo tipo presenta un mayor desafío, ya que los algoritmos deberían contemplar la utilización de cores que varían en rendimiento, consumo energético e incluso en su conjunto de instrucciones.

Obtener aplicaciones con valores altos en métricas como el speedup y la eficiencia requieren sacar el mayor provecho a la arquitectura disponible. Esto significa poder distribuir el trabajo a todos los cores disponibles, evitando que alguno permanezca ocioso. La asignación de trabajo no necesariamente será equitativa; sino que, como se mencionó previamente, las arquitecturas heterogéneas requerirán una distribución necesariamente dispar, con mayor grado de análisis. Asignar el trabajo a los diferentes cores de la arquitectura disponible es lo que se conoce como balance de carga e incidirá directamente en el rendimiento de la aplicación (Naiouf & De Giusti, 2003; Naiouf *et al*. 2006; Willebeek-LeMair & Reeves, 1993).

Existe la necesidad imperiosa de presentar al alumno todos estos conceptos de concurrencia y paralelismo que se han mencionado previamente. No sólo para mantener una currícula vigente y moderna, sino porque efectivamente todas las arquitecturas y sistemas existentes, que utilizarán durante su etapa educativa, son en esencia paralelas. Según sugieren los planes internacionales para el estudio de Ciencias de la Computación e Ingeniería Informática este tipo de contenidos deben ser introducidos en los inicios de la formación del alumno (ACM/IEEE-CS, 2013; ACM/IEEE-CS, 2015). El desafío es tratar de encontrar estrategias y herramientas que hagan posible abordar de forma temprana, conceptos fundamentales de concurrencia que son por su naturaleza complejos.

Contar con herramientas interactivas para la enseñanza de estos temas, en un curso CS1 (cursos introductorios de programación), es algo sumamente deseado (ACM/IEEE-CS, 2004; ACM/IEEE-CS, 2008). El aprendizaje de conceptos de cierta complejidad y un alto grado de abstracción es beneficiado cuando se pueden visualizar dichos conceptos de forma gráfica y en un entorno utilizado en la vida cotidiana. Hay que resaltar que la estimulación a través de dispositivos electrónicos como celulares, tablets, computadoras, etc. es empleada cada vez con mayor frecuencia y en etapas tempranas en la vida del alumno (Hoonlor *et al.* 2013).

Actualmente se dispone de un entorno gráfico CMRE utilizado en curso iniciales de programación de algoritmos (De Giusti *et al.* 2012a; De Giusti *et al.* 2012b). El entorno cuenta con un conjunto de robots que recorren una ciudad interactuando con flores y papeles. En su versión actual se utiliza para presentar los conceptos básicos de la concurrencia, sin embargo, no contempla ciertas características más avanzadas relacionadas principalmente con el paralelismo (De Giusti *et al.* 2015). Entre ellas se pueden mencionar la heterogeneidad, balance de carga, consumo energético y temperatura. El objetivo de este trabajo es extender el entorno existente CMRE a partir de la incorporación de estos nuevos conceptos, de forma gráfica e interactiva, y

que resultan indispensable incluirlos en las nuevas currículas. Se espera que la implementación a realizar colabore en los procesos de enseñanza y aprendizaje de los temas relacionados con la concurrencia y el paralelismo que se abordan en los cursos iniciales de programación, así como también en cursos de años superiores.

1.4 Desarrollos propuestos

En el presente trabajo se propone extender el entorno CMRE (Concurrent Multi Robot Environment) utilizado actualmente en un curso inicial de programación de algoritmos. Se implementarán funcionalidades de manera que cada robot (procesador) pueda trabajar a distinta velocidad y al mismo tiempo se incorpore la información de consumo de energía y temperatura empleados por cada robot.

Por cada operación (instrucción) que realizan los robots (Moverse, Depositar / Recoger / Comunicarse por Mensajes / Informar), se determinará el consumo energético y en qué valor incrementa la temperatura del procesador. Se realizará un relevamiento de las acciones involucradas en cada operación del robot, estableciendo así los valores de consumo y temperatura acordes al esfuerzo requerido. Estos valores recomendados y configurados por defecto podrán ser modificados por el alumno desde la aplicación. Los mismos serán configurables para cada robot, permitiendo de este modo simular las arquitecturas multicore heterogéneas.

A medida que cada robot se encuentre en ejecución, se actualizará su consumo y temperatura, en base a la carga de trabajo y velocidad. La velocidad del robot será un factor a destacar en el cálculo de actualización. El usuario podrá visualizar, durante la ejecución, los cambios producidos de forma gráfica y amigable. Se proveerá un mecanismo para incrementar la velocidad del robot (procesador) cuando su carga de trabajo lo permita, y otro para disminuir la velocidad cuando el robot se encuentre exigido y su temperatura sobrepase los valores normales. La temperatura máxima de cada robot será un atributo editable por el alumno, pero al igual que el costo por instrucción, se brindará un valor estimado en base a especificaciones de procesadores actuales.

Durante la ejecución, cada robot almacenará información de su estado, registrando cambios de velocidad, temperatura, consumo, entre otros. Se desarrollará un módulo para presentarle al alumno dicha información, una vez finalizada la ejecución. El mismo contará con gráficos de distinto tipo (línea, barra, torta) y tablas totalizadoras. El alumno podrá analizar los datos resultantes observando el trabajo realizado por cada robot (procesador), y considerar la

posibilidad de reestructurar los algoritmos, en caso de ser necesario para balancear la carga y así evitar que algún procesador se encuentre ocioso o, por el contrario, sobrecargado.

La implementación del módulo descripto será utilizada en la cátedra de Taller de Programación de la Facultad de Informática, para presentar a los alumnos estos conceptos de forma gráfica e interactiva. A partir de los resultados obtenidos y la respuesta de los alumnos, se podrán plantear posibles mejoras a futuro.

1.5 Estructura de trabajo

En el capítulo 2, se presentan conceptos de concurrencia y paralelismo, profundizando en las arquitecturas multicore. Se describen problemáticas actuales como el incremento en el consumo energético y la temperatura generada por los procesadores contemporáneos. En cuanto a la programación, se analiza la necesidad de crear aplicaciones que exploten los núcleos del procesador y al mismo tiempo resulten eficientes desde el punto de vista energético.

En el capítulo 3, se plantea la necesidad de enseñar al alumno de carreras informáticas los conceptos de concurrencia y paralelismo. Las arquitecturas actuales son en esencia paralelas, y las cátedras deben encontrar instrumentos que faciliten la enseñanza y el aprendizaje. Se exhiben estrategias y herramientas para abordar estos contenidos que revisten cierta complejidad, cuando se incorporan tempranamente en la carrera.

En el capítulo 4, se presenta el entorno CMRE utilizado actualmente en cursos iniciales de programación de algoritmos. Se detalla la funcionalidad en su versión actual, que tiene como objetivo la enseñanza y el aprendizaje de los conceptos básicos de concurrencia de forma gráfica e interactiva. Se definen las primitivas del lenguaje de programación utilizado en CMRE y se presentan algunos ejemplos de uso.

En el capítulo 5, se define un prototipo que presenta una ampliación al entorno CMRE implementando una nueva funcionalidad que tiene como propósito introducir los conceptos de paralelismo relacionados con las arquitecturas multicore, el balance de carga, el consumo energético y la temperatura. Se utilizan ejemplos de algoritmos para analizar los cambios realizados al entorno actual.

En el capítulo 6, se presenta un caso de estudio realizado en el marco de la materia Taller de Programación, aprovechando el conocimiento y la práctica que tienen los alumnos sobre el entorno CMRE. Se presentan los resultados obtenidos a partir de la realización de una actividad conjunta para trabajar los conceptos de temperatura, consumo y velocidad que incorpora ECMRE, y de una encuesta a los alumnos obteniendo así un primer feedback de la aplicación.

En el capítulo 7, se presenta las conclusiones finales obtenidas a partir de la realización de la tesina. Se contrastan los objetivos planteados inicialmente con el trabajo realizado, analizando el cumplimiento de los mismos y el resultado obtenido.

Por último, se detallan posibles trabajos a futuro con el propósito de mejorar y actualizar el entorno ECMRE teniendo presente los cambios tecnológicos de la actualidad.

1.6 Resultados esperados

Del corriente trabajo se esperan obtener los siguientes resultados:

- Un informe sobre conceptos de procesadores multicore y nuevas arquitecturas, focalizando principalmente en el consumo de energía y la generación de calor que se produce en los procesadores al ejecutar aplicaciones.
- Con el diseño e implementación de nuevas funcionalidades incorporadas en el entorno ECMRE (Extended Concurrent Multi Robot Environment) se espera poder colaborar en la enseñanza y el aprendizaje de conceptos complejos de concurrencia y paralelismo, de forma gráfica e interactiva. La implementación será el resultado de las investigaciones realizadas.
- Obtención de un documento a partir de un caso de estudio con alumnos de la cátedra de Taller de Programación de la Facultad de Informática, dónde se presentan los resultados obtenidos para una posterior evaluación. Ese documento se espera que refleje un análisis objetivo, en relación a la utilización del entorno ECMRE extendido en el ámbito educativo.

CAPITULO 2
ARQUITECTURAS PARALELAS

2.1 Definiciones y Conceptos Generales

En el capítulo 1 se ha propuesto la realización de una ampliación funcional al entorno de enseñanza CMRE (Concurrent Multi Robot Environment) utilizado en la Facultad de Informática de la UNLP. Esta extensión llamada ECMRE (Extended Concurrent Multi Robot Environment) incorpora nuevos conceptos de concurrencia y paralelismo, entre los que se encuentran los procesadores multicore y sus arquitecturas, el consumo de energía, temperatura de los procesadores, balance de carga de la aplicación, entre otros. En el presente capítulo, se estudian los conceptos incorporados al entorno ECMRE basados en el componente físico central y más complejo de un sistema informático, denominado procesador. En aquellos casos donde el procesador integra dos o más núcleos computacionales dentro de un mismo "chip" es llamado multicore. El procesador es el encargado de ejecutar los programas (Tinetti & De Giusti, 1998). La ejecución de un programa secuencial con un sólo flujo de control es conocido como proceso; en el cual se ejecuta una instrucción y cuando ésta finaliza ejecuta la siguiente. Cada proceso puede residir en un procesador independiente o dedicado. También se pueden tener múltiples procesos sobre el mismo procesador (Tinetti & De Giusti, 1998). Los procesadores actuales tienen la capacidad (denominada concurrencia) de ejecutar múltiples actividades en paralelo o simultáneamente. Un programa concurrente especifica dos o más procesos que cooperan para realizar una tarea. Un sistema concurrente puede disponer de M procesadores, donde cada uno puede ejecutar uno o más procesos (Andrews, 2000). El paralelismo es la ejecución concurrente (en el mismo instante de tiempo) sobre diferentes componentes físicos (procesadores). El paralelismo es un concepto asociado con la existencia de múltiples procesadores ejecutando un algoritmo en forma coordinada y cooperante. Al mismo tiempo se requiere que el algoritmo admita una descomposición en múltiples procesos ejecutables en diferentes procesadores (concurrencia). Cuando el sistema de hardware está formado por un conjunto de procesadores o elementos de procesamiento vinculados, como por ejemplo una red, con capacidad de ejecutar coordinadamente un algoritmo general, se obtiene paralelismo. Una arquitectura paralela es el soporte de hardware para poder tener procesamiento concurrente real (Tinetti & De Giusti, 1998).

2.2 Necesidad de Cómputo Paralelo

Hasta aquí se han definido brevemente los conceptos de concurrencia y paralelismo. La idea de poder ejecutar múltiples actividades en paralelo, o simultáneamente, existe desde hace décadas, y es un concepto planteado como una necesidad aun cuando las arquitecturas de hardware no vislumbraban la posibilidad de lograrlo. En el año 1945, Von Neumann (Von Neumann, 1945; Burks *et al.*, 1946) presentó un modelo que describe una arquitectura de computadora con una unidad de procesamiento, una memoria donde almacenar datos e instrucciones y mecanismos de entrada y salida. Desde su nacimiento, la arquitectura inicial ha evolucionado, aunque siempre asumiendo la disponibilidad de un procesador con un único hilo de control. La concurrencia sólo podía ser explotada parcialmente al ejecutar un algoritmo y el paralelismo se encontraba limitado por la tecnología de hardware disponible (De Giusti *et al.*, 2015). Aun así, la búsqueda del paralelismo se mantuvo vigente debido a la existencia de necesidades informáticas que requerían solución. Entre ellas se encontraban:

- Algunos problemas de procesamiento sólo pueden ser tratados mediante el cómputo paralelo (Akl, 1989). Numerosas aplicaciones requieren manipular grandes cantidades de datos en un intervalo de tiempo muy corto. Aquí la velocidad es crucial y, en principio, no es posible de lograr con una arquitectura monoprocesador.
- Disponer de cantidades masivas de datos en Internet, tuvo como consecuencia la aparición de nuevas aplicaciones que trabajan con esta inmensa cantidad de datos (Hennessy & Patterson, 2012).
- Incrementar el interés y la necesidad de contar con servidores de gama alta, como por ejemplo el Cloud Computing y el Software como Servicio (Software as a Service, SaaS) (Hennessy & Patterson, 2012).

A causa de las menores eficiencias en el uso de silicio y energía que se encontraron entre 2000 y 2005, los diseñadores trataron de encontrar y explotar más posibilidades de paralelismo a nivel de instrucción (ILP), algo que resultó ser ineficiente, ya que los costos de energía y silicio crecieron, en relación, más rápido que el rendimiento. Aparte de ILP, la única forma escalable y de propósito general que se conoce para aumentar el rendimiento más rápido de lo que la tecnología básica permite es mediante multiprocesamiento. La posibilidad de que una computadora pueda realizar multiprocesamiento estuvo restringida a la ocurrencia de un hito posterior al nacimiento de las mismas, que fue el surgimiento de los procesadores multicore. A continuación, se describe someramente la evolución de los procesadores para obtener un mayor poder de cómputo y se analizan las principales limitaciones encontradas por los arquitectos de hardware. Por otra parte, se presentan algunas alternativas empleadas para sortear tales

limitaciones hasta alcanzar el diseño del procesador multicore o multinúcleo que se conoce actualmente.

2.3 Surgimiento de procesadores Multicores

A partir del nacimiento de los microprocesadores en 1971, la industria ha continuado con éxito la innovación y el aumento de rendimiento.
Según Gepner & Kowalik (2006), existen varias alternativas para mejorar el rendimiento de un procesador:

a. Tecnologías de procesos más sofisticadas: este punto involucra al proceso de diseño de circuitos semiconductores en silicio y a las metodologías de fabricación utilizadas para crear transistores cada vez más pequeños, más rápidos y más eficientes en energía. El resultado es la creación de un chip más sofisticado e integrado.
b. Arquitectura innovadora: comprende al conjunto de instrucciones, registros y estructuras de datos que son públicas para el programador y se mantienen y mejoran de una generación a la siguiente.
c. Micro-arquitectura: se refiere a una implementación de la arquitectura del procesador en silicio.

El rendimiento de un procesador combina la frecuencia de reloj y las instrucciones por ciclo del reloj. Se define como:

$$\boldsymbol{Rendimiento = Instrucciones\ ejecutadas\ por\ ciclo * Frecuencia} \quad \textit{(Fórmula 2.1)}$$

Durante mucho tiempo, los arquitectos de hardware incrementaron la frecuencia del reloj y el número de transistores en el chip para aumentar la potencia de un procesador. La frecuencia es una función tanto del proceso de fabricación como de la microarquitectura (Gepner & Kowalik, 2006). De acuerdo a la Ley de Moore, el número de transistores en el chip de un procesador se duplica cada 18-24 meses (Moore, 1965). Tal como lo predijo Moore, la densidad de transistores por unidad de superficie, fue aumentando año a año predominando sobre el manejo de consumo. Un chip con millones de transistores a poca distancia ayuda a incrementar drásticamente el rendimiento, pero a su vez aumenta la fuga de corriente y la cantidad de calor generada por el chip.

El otro gran objetivo que han perseguido, los diseñadores, es la búsqueda de una ejecución concurrente real (paralelismo). Como se mencionó previamente existen numerosas razones que demandan una arquitectura paralela. Algunas han nacido o tomado mayor relevancia en las últimas dos décadas, sin embargo, desde los inicios del procesador se planteaba un salto en

rendimiento al lograr la ejecución simultánea de instrucciones. Una primera aproximación fue el paralelismo a nivel de instrucción (ILP), y consiste en que el procesador pueda analizar las instrucciones a ejecutar, y su dependencia de datos. De esta forma, es posible ejecutar instrucciones en paralelo que no dependen entre sí (Eijkhout *et al.*, 2015).

La búsqueda de un incremento de rendimiento en los procesadores y la ejecución paralela, trajo consigo la implementación de técnicas avanzadas de procesamiento. A continuación, se describen las principales técnicas que han sido utilizadas: (Grama *et al.*, 2003; Tinetti & De Gisti, 1998; Rucci, 2015; Hager & Wellein, 2011)

- Pipelining

  El procesamiento pipelining aprovecha la filosofía de trabajo en cadena, donde cada etapa completa una parte de la tarea total. Consiste en dividir las instrucciones en partes (segmentos), para que el procesador pueda ejecutar simultáneamente segmentos de distintas instrucciones. El resultado es un incremento en el número de instrucciones ejecutadas por ciclo de reloj. Este es el ejemplo más elemental de paralelismo a nivel de instrucciones.
- Arquitectura superescalar

  Los procesadores superescalares incorporan múltiples unidades funcionales, capaces de ejecutar más de una instrucción por ciclo de reloj. Este tipo de arquitecturas posee varios pipelines segmentados, que pueden iniciar la ejecución de varias instrucciones al mismo tiempo. El orden de ejecución puede ser distinto al impuesto por el programa original. Para ello el procesador posee un planificador a nivel de hardware que analiza el programa a ejecutar y extrae aquellas instrucciones que pueden ser ejecutadas simultáneamente.
- SIMD (Single Instruction Multiple Data)

  Estas arquitecturas permiten ejecutar la misma operación sobre un conjunto de datos diferentes al mismo tiempo. Poseen un conjunto de elementos de procesamiento iguales, que son administrados por una sola unidad de control. La ejecución es sincrónica (se dispone de un reloj global) y los elementos de procesamiento pueden comunicarse utilizando memoria compartida o una red de comunicaciones.
- OoOE (Out of Order Execution)

  La ejecución fuera de orden es un paradigma utilizado por los procesadores de alto rendimiento para aprovechar ciclos de instrucción que podrían desperdiciarse (si se ejecuta el algoritmo en orden). Las instrucciones son reorganizadas teniendo en cuenta dependencias, y como resultado se obtiene un orden de ejecución más eficiente al original.

- Ejecución especulativa y predicción de saltos

  La ejecución especulativa consiste en que el procesador ejecute una porción de código que aún no es necesaria, pero se estima que lo será próximamente. La predicción de saltos predice cual es la próxima instrucción a ejecutar, asumiendo el valor de verdad de futuras instrucciones condicionales. En ambos casos, si se ha acertado se continúa con la ejecución (habiendo adelantado el trabajo). Por el contrario, si la especulación ha fallado, se deshacen los cambios y se descarta el trabajo realizado.
- SMT (Simultaneous Multithreading)

  Esta técnica proporciona al sistema operativo dos o más procesadores lógicos, aunque físicamente dispone de un único procesador. Para ello, se requiere duplicar algunos componentes de hardware que permiten almacenar el contexto de cada procesador lógico. El resultado es la ejecución paralela de múltiples flujos de instrucciones (también llamados hilos hardware). Los procesadores Intel, han utilizado esta técnica con el nombre comercial Hyper-Threading.
- Incremento de memorias caché:

  La técnica de aumentar el tamaño de la memoria caché busca incrementar el rendimiento, reduciendo el tiempo de acceso a los datos ubicados en la memoria principal.

A partir del año 2000, el proceso de mejora en los procesadores se estancó principalmente por dos razones: (Eijkhout *et al.*, 2015)

- No es posible seguir aumentando la frecuencia del procesador debido a que la generación de calor y consumo de energía afecta el funcionamiento de los circuitos integrados. Esta limitación impuesta por el problema energético se conoce como *power wall*.
- La tarea de obtener mayor paralelismo a nivel de instrucción (ILP) en los programas fue cada vez más compleja, por los siguientes motivos:
  - Limitaciones impuestas por los compiladores utilizados.
  - Limitaciones en la cantidad de paralelismo intrínseco disponible.
  - La predicción de saltos hacía imposible incrementar el ILP.

La solución que encontraron los arquitectos de hardware para poder incrementar el rendimiento sorteando estos impedimentos, fue integrar más de un núcleo computacional dentro de un mismo chip. Este chip es conocido como procesador multicore o multinúcleo. Los procesadores multicore mejoran el rendimiento de una aplicación al distribuir el trabajo entre los núcleos disponibles, reduciendo el consumo de energía en cada núcleo. Dos núcleos a una frecuencia más baja pueden tener el mismo rendimiento que un único procesador a una frecuencia más alta;

por lo tanto, múltiples núcleos son más eficientes en energía. El Paralelismo a nivel de instrucción es reemplazado por un paralelismo de tareas explícito y gestionado por el programador. (McCool, 2008; Eijkhout *et al.*, 2015). Los procesadores multinucleos han sido combinados para formar tipos de arquitecturas multicores que poseen características propias y ofrecen al usuario final distintas prestaciones.

2.4 Arquitecturas Multicores

La arquitectura de los procesadores actuales es paralela. En base a las características de sus núcleos existen dos tipos de multicores: (De Giusti et al., 2015; Hyari, 2009; Rao & Bala Krishna, 2013; Naiouf *et al.*, 2013)

- Arquitecturas Multicores Homogéneas

  Todos sus cores poseen la misma arquitectura, es decir, tienen las mismas características. Manejan exactamente los mismos tamaños de caché, comparten el espacio de memoria (o al menos una parte) y compiten en igualdad de condiciones por dicho acceso. Los microprocesadores se comunican con la memoria mediante un bus compartido (recurso de uso común). Normalmente utilizan un único sistema operativo que se ejecuta en todos los cores. Cada core realiza una ejecución diferente sobre un conjunto distinto de datos, pero con la capacidad de compartir recursos comunes (memoria, dispositivos de E/S, sistema de interrupciones, entre otros) conectados al bus del sistema. La simplicidad de su diseño, hace que sea relativamente fácil de implementar y a bajo costo. Proporciona un gran rendimiento para aquellas aplicaciones con alto grado de paralelización, junto con una sencilla gestión de recursos compartida.
- Arquitecturas Multicores Heterogéneas

  Poseen cores con diferentes características en cuanto a rendimiento y consumo de energía, pudiendo utilizar o no distintos conjuntos de instrucciones (ISA: *instruction set architecture*). Cada core tiene su propio espacio de direcciones y puede ejecutar un sistema operativo distinto. Se proporciona algún mecanismo para facilitar la comunicación entre las CPUs. Disponer de cores de diferentes tipos posibilita optimizar el rendimiento de las aplicaciones ejecutadas en la arquitectura paralela. El resultado de distribuir correctamente la carga de trabajo en los núcleos, provoca una mayor eficiencia en la relación rendimiento/energía. Como desventaja principal se puede mencionar que cuando se utilizan distintos conjuntos de instrucciones, el código fuente de un programa debe ser compilado para cada tipo de core.

Ambas arquitecturas multicore han significado un salto generacional en cuestiones de procesamiento paralelo, incremento de rendimiento, mejoras en consumo energético y

generación de calor. La elección del tipo de arquitectura dependerá de cada escenario particular. Las arquitecturas homogéneas se adecuan a aquellas aplicaciones con alto grado de paralelización, ya que la existencia de código secuencial disminuye la *performance* del sistema. Por otro lado, una arquitectura heterogénea provee un mejor rendimiento y manejo energético, cuando los núcleos son distribuidos correctamente. Contar con cores de diferentes características, brinda flexibilidad para resolver problemas del mundo real que son en su esencia heterogéneos.

2.4.1 Rendimiento en arquitecturas paralelas

El rendimiento de aplicaciones paralelas sobre arquitecturas multicore, se analiza utilizando métricas tradicionales como speedup y eficiencia (De Giusti *et al.*, 2015). Antes de definir estas métricas es necesario presentar algunos conceptos introductorios: (Grama *et al.*, 2013)

- Tiempo de ejecución serial: es el tiempo transcurrido para ejecutar un programa sobre una computadora secuencial.
- Tiempo de ejecución paralelo: es el tiempo transcurrido para ejecutar un programa sobre una computadora paralela. Esto es, el intervalo de tiempo desde que se inicia la ejecución, hasta que termina el último elemento de procesamiento.

Speedup (S)

Es una medida para determinar cuál es el beneficio relativo de resolver un problema en paralelo. Esto es, cuan "rápido" ejecuta el algoritmo paralelo respecto al secuencial. La función de speedup se define como el cociente entre el tiempo de ejecución del mejor algoritmo serial conocido (sobre un simple núcleo) y el tiempo de ejecución para resolver el mismo problema de forma paralela (sobre una arquitectura multi-núcleo) (Andrews, 2000; De Giusti, 2008; De Giusti *et al.*, 2015; Grama *et al.*, 2013).

$$\boldsymbol{S = \frac{T_s}{T_p}} \qquad \textit{(Fórmula 2.2)}$$

Eficiencia (E)

Es una medida de la porción de tiempo en la que los cores son usados útilmente en la aplicación paralela. Los procesadores no se encuentran el 100% de su tiempo realizando cómputo, y la eficiencia determina la fracción de tiempo en la que son útiles. Establece la relación entre el speedup obtenido y el speedup óptimo (Sopt) que se puede conseguir en la arquitectura.

$$\boldsymbol{E = \frac{S}{S_{opt}}} \qquad \textit{(Fórmula 2.3)}$$

En arquitecturas homogéneas, el speedup óptimo está dado por la cantidad de cores utilizados (p).

$$\boldsymbol{E = \frac{S}{p}} \qquad \textit{(Fórmula 2.4)}$$

En arquitecturas heterogéneas, se debe considerar la potencia de cómputo de los diferentes núcleos que la componen, por lo cual se define el speedup óptimo como:

$$\boldsymbol{S_{opt} = \sum_{i=0}^{p-1} \frac{Potencia\ (Core_i)}{Potencia\ (Mejor\ Core)}} \qquad \textit{(Fórmula 2.5)}$$

En consecuencia, la eficiencia en una arquitectura heterogénea es:

$$\boldsymbol{E = \frac{S}{\sum_{i=0}^{p-1} \frac{Potencia\ (Core_i)}{Potencia\ (Mejor\ Core)}}} \qquad \textit{(Fórmula 2.6)}$$

El valor que determina la eficiencia es número dentro del rango 0 y 1, y depende del grado de productividad que han tenido los procesadores involucrados durante la ejecución. Cuando la eficiencia es 1, se ha obtenido un speedup perfecto (Andrews, 2000; De Giusti, 2008; De Giusti *et al.*, 2015; Grama *et al.*, 2013).

Overhead Paralelo Total (To)

Como se mencionó previamente, dado un algoritmo, los procesadores de una arquitectura paralela no se encuentran realizando cómputo en todo momento. Esto se debe a factores como la existencia de componentes secuenciales en la ejecución, tiempos de sincronización y comunicación, entre otros. El overhead paralelo total es la cantidad de tiempo insumido por todos los elementos de procesamiento al ejecutar el algoritmo de forma paralela, que se encuentra por encima del tiempo más rápido conocido para resolver el mismo problema de forma serial (Andrews, 2000; De Giusti, 2008; Grama *et al.*, 2013).

$$\boldsymbol{T_o = pT_p - T_s} \qquad \textit{(Fórmula 2.7)}$$

Costo

El costo de resolver un problema en un sistema paralelo refleja la suma del tiempo de cada procesador que se encuentra realizando cómputo. Se define como el producto entre el tiempo de ejecución paralelo (T_p) y la cantidad de elementos de procesamiento utilizados (p).

$$\boldsymbol{Costo = p\ T_p} \qquad \textit{(Fórmula 2.8)}$$

Un sistema es óptimo en cuanto a costo si el costo de resolver un problema en el sistema paralelo tiene el mismo crecimiento asintótico (en función del tamaño del problema), que el

costo en el mejor sistema secuencial equivalente. Un sistema paralelo de costo óptimo tiene una eficiencia de O(1), ya que definición de eficiencia es el cociente entre el costo secuencial y el costo paralelo (Andrews, 2000; De Giusti, 2008; Grama *et al.*, 2013).

Grado de Concurrencia (C(W))

Establece el mayor número de tareas que pueden trabajar al mismo tiempo y durante toda la ejecución del algoritmo. Dado un tamaño de problema (denominado W), el grado de concurrencia indica que no se podrán usar más de C(W) procesadores. Su valor dependerá del algoritmo involucrado, ya que cada problema a resolver tiene un grado de paralización diferente. El grado de concurrencia no hace un análisis de los procesadores y recursos disponibles, por lo tanto, C(W) puede no ser viable en una computadora real (Andrews, 2000; De Giusti, 2008; Grama *et al.*, 2013).

2.4.2 Técnicas de ajuste de rendimiento en procesadores

Como se mencionó anteriormente el rendimiento de los procesadores es evaluado a partir de la utilización de un conjunto de métricas como son el speedup y la eficiencia. En los últimos años los procesadores son diseñados con la capacidad de aumentar/disminuir su velocidad en base a ciertos parámetros del sistema y de las necesidades de cómputo de las aplicaciones. A partir de esta característica surgen dos técnicas llamadas Overclocking y Underclocking que afectan la *performance* global del sistema y en consecuencia al resultado de aplicar las métricas de rendimiento.

2.4.2.1 Overclocking

Es un proceso utilizado para lograr que un componente electrónico funcione a una velocidad superior que la determinada por el diseño y especificaciones del fabricante. El objetivo es obtener un rendimiento mayor en sistemas de alta performance, aplicando esta técnica a procesadores, tarjetas gráficas, chips de la placa madre, memorias, entre otros. Al utilizar overclocking en un procesador se incrementa la frecuencia del reloj (velocidad), y se obtiene una ganancia en rendimiento, lo que aumenta la performance global del sistema.

Incrementar la frecuencia del reloj, resulta inevitablemente en un incremento del consumo de corriente, y en consecuencia del calor generado. En la actualidad, los fabricantes de hardware contemplan un uso moderado de overclocking, ya que al explotarlo a gran escala ocurren situaciones indeseables como:

- Necesidad de refrigeración: el incremento de calor generado por el aumento de corriente puede tener efectos adversos. En primera instancia requiere de un sistema de enfriamiento eficaz, siendo necesario en algunos casos mejorar la refrigeración de fábrica (por ejemplo, mediante la utilización de disipadores de calor y ventiladores).
- Inestabilidad del sistema: un procesador expuesto a temperaturas superiores a sus especificaciones comienza a causar fallas de funcionamiento, que van desde un error esporádico y la generación de incorrecta de datos, hasta el bloqueo total del sistema.
- Daños físicos: la exposición continua de los procesadores a altas temperaturas puede dañar el hardware. Algunos fabricantes contemplan el uso de overclocking, sin embargo, no incluyen en la garantía los daños producidos por su utilización.

2.4.2.2 Underclocking

Underclocking es la técnica opuesta a overclocking y consiste en reducir la frecuencia de reloj de un componente. Se utiliza en procesadores para obtener ventajas en el funcionamiento del sistema, entre las que se puede mencionar:

- Disminución de temperatura: al reducir la velocidad del procesador, se logra una disminución de la temperatura beneficiando la refrigeración del sistema.
- Reducción de voltaje: permite obtener ahorros de electricidad.
- Incrementar la vida útil: cuando el sistema funciona a bajas temperaturas y sin grandes exigencias, incrementa su durabilidad.

Dado que underclocking disminuye el rendimiento del sistema, debe ser utilizado en escenarios donde la carga de trabajo esté por debajo de la capacidad disponible (Uht & Vaccaro, 2004; Lang & Patel, 2009).

En sus inicios, el overclocking y underclocking fueron implementados de forma manual por los usuarios más especializados, alterando los valores por defecto del sistema BIOS (Basic Input Output System) y asumiendo la responsabilidad ante las consecuencias que esto podía causar. En la actualidad los fabricantes de procesadores contemplan y proveen tecnologías para alterar la velocidad del procesador, dinámicamente, en base a la carga de trabajo. A modo de ejemplo se describen, a continuación, las tecnologías utilizadas por Intel y AMD.

Intel ha trabajado en la fabricación de procesadores con nuevas tecnologías que permiten obtener un mejor desempeño cuando sea necesario. La tecnología Turbo Boost se encuentra en plena vigencia y se utiliza en procesadores Intel® Core™ i3 de sexta generación, procesadores

Intel® Core™ i5 de sexta generación, procesadores Intel® Core™ i7 de sexta generación y procesadores de PC Intel® de alta gama, entre otros.

- Tecnología Intel Turbo Boost: permite incrementar de forma automática la velocidad del procesador mientras lo permitan los valores de consumo de energía, temperatura y TDP (Intel, 2017a).
- Tecnología Intel Turbo Boost 2.0: la versión 2.0, introduce nuevas eficiencias de energía utilizando un chip integrado para el procesador y los gráficos. Incrementa de forma automática, la velocidad de procesamiento de los núcleos del procesador por encima de la frecuencia operativa nominal. El grado de incremento se encuentra fuertemente relacionado y acotado por los siguientes factores:

  - ✓ Tipo de carga de trabajo
  - ✓ Cantidad de núcleos activos
  - ✓ Consumo estimado de corriente
  - ✓ Consumo estimado de energía
  - ✓ Temperatura del procesador

  Cuando el procesador se encuentra funcionando por debajo de sus límites y la carga de trabajo del usuario exige mayor desempeño, se aumenta dinámicamente la frecuencia del procesador hasta alcanzar su límite superior. La tecnología Intel® Turbo Boost 2.0 posee varios algoritmos que funcionan en paralelo para administrar la corriente, energía y temperatura, a fin de maximizar la frecuencia y la eficiencia energética. Al mismo tiempo, permite que el procesador funcione a un nivel de energía mayor que su TDP (Thermal Design Power) configurada y la energía indicada en sus especificaciones durante períodos breves, a fin de maximizar el desempeño. La Tabla 2.1 es una versión reducida, que presenta cómo afecta esta tecnología en algunos procesadores Intel Desktop i5 e i7 de sexta generación (Intel, 2017b; Intel, 2017c).

Matriz de Intel Turbo Boost Technology 2.0												
6ta generación Intel Core i5 Desktop Procesadores												
Procesador	**i5-6685R, 3,20 GHz**				**i5-6585R, 2,80 GHz**				**i5-6600T, 2,70 GHz**			
Núcleos de procesador	Quad-core				Quad-core				Quad-core			
Núcleos activos	4C	3C	2C	1C	4C	3C	2C	1C	4C	3C	2C	1C
Bin de máxima de Turbo Boost tecnología	1	3	5	6	3	5	7	8	6	6	7	8
Frecuencia máxima de Turbo Boost	3.3	3.5	3.7	3.8	3.1	3.3	3.5	3.6	3.3	3.3	3.4	3.5
6ta generación Intel® Core™ i7 Desktop Procesadores												
Procesador	**I7-6785R – 3,30 GHz**				**I7 - 6700K - 4.00 GHz**							
Núcleos de procesador	Quad-core				Quad-core							
Núcleos activos	4C	3C	2C	1C	4C	3C	2C	1C				
Bin de máxima de Intel® Turbo Boost tecnología	2	4	5	6	0	0	0	2				
Frecuencia máxima de Intel® Turbo Boost	3.5	3.7	3.8	3.9	4	4	4	4.2	**1 bin = 100 MHz**			

Tabla 2.1: Tecnología Turbo Boost 2.0 en procesadores Intel (extraída de la página oficial de Intel).

- Tecnología Intel Turbo Boost Max 3.0: es la tecnología más reciente de Turbo Boost, disponible en los últimos diseños de procesadores. Su objetivo no es reemplazar a la tecnología Intel® Turbo Boost 2.0, sino mejorarla con un incremento extraordinario en la frecuencia de su núcleo más veloz (aprovechando al máximo el procesador). Permite identificar y dirigir las cargas de trabajo al núcleo más rápido del chip. Utiliza un controlador junto con la información almacenada en la CPU para identificar cuál es el mejor núcleo disponible y asignar el trabajo más crítico hacia él. Su funcionamiento depende de varios factores: (Intel, 2017d)

    - ✓ Tipo de carga de trabajo
    - ✓ Cantidad de núcleos activos
    - ✓ Consumo estimado de corriente
    - ✓ Consumo estimado de energía
    - ✓ Temperatura del procesador
    - ✓ Compatibilidad de unidades

Por su parte AMD cuenta con una tecnología equivalente de overclocking denominada Tecnología Turbo Core. Su objetivo es ofrecer un mayor rendimiento del sistema cuando se requiere un alto rendimiento del procesador. Para lograrlo, la frecuencia del procesador se incrementa de forma dinámica hasta alcanzar el límite superior de frecuencia soportado. Una vez que la carga de trabajo se reduce, el núcleo vuelve a disminuir su frecuencia hasta valores

normales. El incremento de velocidad, en el procesador, se encuentra acotado por los límites de potencia y temperatura (AMD, 2017).

Se han presentado diversas tecnologías utilizadas en procesadores multicore actuales para alterar la velocidad del procesador dinámicamente en base a la carga de trabajo. Un factor importante para aprovechar al máximo la capacidad de cómputo del sistema es que no existan núcleos ociosos mientras otros están ejecutando. Esto dependerá en gran parte de cómo se distribuye la carga de trabajo entre los núcleos disponibles, lo que se conoce como balance de carga.

2.4.3 Balance de carga

El objetivo principal del cómputo paralelo consiste en resolver un problema reduciendo los tiempos de ejecución, mientras que los recursos son utilizados de forma eficiente. Para cumplir este objetivo, se debe distribuir la carga de trabajo (tareas) entre todos los procesadores disponibles para que ejecuten durante una misma cantidad de tiempo (o tan aproximada como sea posible). La acción de mapear las tareas a procesadores de forma equilibrada es lo que se conoce como balance de carga. Notar que una ejecución desbalanceada implica que uno o más procesadores estarán ociosos, esperando a que otros terminen su ejecución; y esto a su vez genera una disminución en los valores de las métricas como speedup y eficiencia (Naiouf *et al.*, 2011; De Giusti *et al.*, 2015). El balance de carga intenta alcanzar dos objetivos: (Grama *et al.*, 2013)

- Reducir la cantidad de tiempo que gastan los procesos para interactuar unos con otros.
- Reducir el tiempo total de los procesos que se encuentran inactivos, mientras que otros realizan parte del cómputo.

El tipo de arquitectura multicore es un factor que incide directamente sobre el balance de carga. Cuando todos los procesadores tienen las mismas características (arquitecturas multicores homogéneas), la asignación de tareas es relativamente sencilla. Idealmente, el grado de concurrencia debería ser igual al número de procesadores disponibles y la cantidad de trabajo distribuida en partes de igual tamaño. Sin embargo, cuando las características de los procesadores difieren (arquitecturas multicores heterogéneas), el balance de carga se convierte en una tarea ardua. En este caso, es necesario contemplar las prestaciones de cada procesador (capacidad de cómputo, memoria, caché, entre otras) y la comunicación entre ellos (Naiouf *et al.*, 2011; De Giusti, 2008; De Giusti *et al.*, 2015a).

Las técnicas de mapeo (asignación de tareas a procesadores), se clasifican en dos grupos: (Grama *et al.*, 2013; De Giusti, 2008; Mesa, 2009)

- Mapeo Estático

  Las técnicas de mapeo estático realizan la distribución de tareas antes de la ejecución del algoritmo. Obtener una asignación óptima es un problema complejo y la asignación estática requiere un conocimiento previo sobre la cantidad de las tareas necesarias, el volumen de datos asociado a cada tarea, su grado de interacción, el paradigma de programación paralela, entro otros. La necesidad de conocer ciertos datos con anterioridad se convierte en la principal desventaja, considerando que la disponibilidad de los mismos no es habitual. Cuando no se cuenta con dicha información, un mapeo estático puede causar grandes desequilibrios de carga. Normalmente, las técnicas de mapeo estático se utilizan (obteniendo buenos resultados) en algoritmos que son fáciles de diseñar e implementar.

- Mapeo Dinámico

  Las técnicas de mapeo dinámico realizan la distribución de tareas durante la ejecución del algoritmo. Existen diferentes escenarios donde el mapeo dinámico ofrece mejores resultados: 1- la cantidad de tareas es superior al número de procesadores, 2- la cantidad de tareas se desconoce, 3- las tareas son generadas dinámicamente. Un factor ha analizar se relaciona con la cantidad de datos de las tareas en relación al cálculo, ya que la asignación dinámica puede implicar el movimiento de esos datos entre procesos. En algunos casos el costo de mover los datos puede significar la conveniencia del uso de una técnica de mapeo estática. Normalmente, las técnicas de mapeo dinámico se utilizan en algoritmos complejos.

2.4.4 Consumo energético

Hasta aquí se han estudiado las arquitecturas paralelas haciendo énfasis en la capacidad de cómputo de las mismas. Durante mucho tiempo el diseño de los sistemas de cómputo ha tenido como único objetivo incrementar la velocidad de procesamiento. El progreso tecnológico en informática ha permitido alcanzar soluciones a problemas de la vida real y acelerar soluciones existentes. Sin embargo, en la última década ha tomado un rol significativo el consumo energético que tienen los sistemas, en particular aquellos de altas prestaciones. Las computadoras de altas prestaciones utilizaron durante mucho tiempo métricas orientadas sólo al rendimiento, como por ejemplo la cantidad de operaciones de coma flotante por segundo (FLOPS). El proyecto TOP500 (iniciado en 1993) realiza un ranking semestral de las 500 supercomputadoras con mayor rendimiento y utiliza FLOPS como métrica de evaluación. Con el tiempo estas supercomputadoras han crecido a tal punto que llegan a consumir el equivalente en electricidad de una ciudad (Markoff & Lohr, 2002). El consumo energético no se corresponde únicamente con el cómputo realizado; gran parte del gasto se utiliza para poder

refrigerar el sistema y asegurar un funcionamiento correcto. Un sistema que sobrepasa los límites máximos de temperatura genera fallos. El consumo energético tiene un costo ambiental y económico. Algunas estimaciones indican que el gasto energético de una supercomputadora en 4 años alcanza el costo del equipo. Esta limitación energética que afecta a los diseñadores de hardware es conocida como *power wall*. El incremento en cómputo se encuentra sujeto a la capacidad de refrigeración del sistema, y la utilización de nuevas técnicas de refrigeración eleva significativamente el costo del producto (Balladini *et al.*, 2013; Montes de Oca *et al.*, 2013; TOP500, 2016).

La llegada de los procesadores multicores ha permitido obtener dos beneficios principales. Por un lado, se obtiene un incremento de cómputo, y al mismo tiempo se reduce el consumo energético. Notar que, en lugar de continuar incrementando la potencia de un procesador con un único núcleo, se utiliza un procesador multicore conformado por más de un core simple, que trabajan a menor frecuencia disminuyendo el consumo, la temperatura generada y el costo de refrigeración. A pesar de esta mejora, la cantidad de energía consumida es un aspecto fundamental, a tener en cuenta, tanto por los diseñadores de hardware, como así también por los programadores que desarrollan aplicaciones. La alta demanda energética tiene graves consecuencias medioambientales y financieras, entre otras. Este concepto se debe tener siempre presente para hacer un uso responsable y eficiente del consumo, reduciendo aquellos gastos innecesarios (Montes de Oca *et al.*, 2013; Naiouf *et al.*, 2015). En la última década, el consumo energético se ha posicionado como uno de los temas más relevantes, encuadrándose bajo el nombre de Green Computing o Green IT.

2.4.4.1 Green Computing

El término Green Computing hace referencia al uso eficiente de los recursos informáticos. Su objetivo es promover el uso de tecnologías que reduzcan el consumo energético y la contaminación ambiental. Intenta desarrollar productos ecológicos, que contribuyan a procesos de reciclado y biodegrabilidad (Schneider Electric, 2008). El consumidor generalmente busca velocidad de cómputo y buenos precios, sin analizar demasiado el impacto ambiental. Sin embargo, el consumo energético se ha incrementado notablemente generando un gasto económico que debe ser evaluado al adquirir un sistema. En sistemas de altas prestaciones el gasto económico asociado al consumo energético puede condicionar desde la escalabilidad del equipo hasta el tipo de aplicaciones que se pueden desarrollar en los mismos. Los diseñadores avanzan en la creación de sistemas que administren la energía, y que dada una aplicación brinden alternativas de ejecución en base a la capacidad eléctrica y de refrigeración disponible,

el consumo energético y la performance requerida (Balladini *et al.*, 2011; Montes de Oca *et al.*, 2013).

Existe un grupo de tecnologías surgidas en el último tiempo que promueven y contemplan un uso eficiente de los recursos informáticos que se describen a continuación: (Maciá, 2006; Naiouf *et al.*, 2015; Sopeña *et al.*, 2008)

- Virtualización

  Consiste en la abstracción de un recurso tecnológico, desde una plataforma de hardware, un sistema operativo o un dispositivo de almacenamiento, entre otros. El recurso en cuestión es creado y administrado por una capa de software. La creación, configuración y eliminación de recursos de forma dinámica y bajo demanda contribuye al uso eficiente de los sistemas. La consolidación del modelo de computación en la nube (cloud computing) ha valorizado esta técnica existente desde el año 1960.
- Cloud Computing

  Un proveedor proporciona grandes conjuntos de recursos físicos y lógicos (como por ejemplo infraestructura, plataformas de desarrollo, almacenamiento, entre otros). Un consumidor (generalmente una empresa), contrata y utiliza recursos mediante una interface de administración web, con un modelo de arquitectura "virtualizada".

  Estos recursos son proporcionados como servicios ("as a service") y pueden ajustarse dinámicamente en base a las necesidades de consumidor, lo que resulta en una técnica altamente escalable. Los costos económicos se adecuan a las prestaciones solicitadas, siendo una alternativa muy interesante para pequeñas y medianas empresas. La utilización de un sistema configurado con recursos acotados pero suficientes (sin sobredimensionar el hardware necesario), beneficia la utilización de los recursos informáticos.
- Computación Grid

  Permite mediante el uso de redes de comunicación a alta velocidad (internet), compartir recursos (CPU, memoria, bases de datos, entre otros) a gran escala, que se encuentran distribuidos geográficamente. Surgió como una iniciativa del ámbito científico y académico, tras la necesidad de contar con grandes capacidades de cálculo computacional y almacenamiento de datos. Actualmente ha evolucionado para dar soporte a las necesidades empresariales. La computación Grid, provee la funcionalidad básica para compartir y aprovechar recursos de diferentes organizaciones de forma transparente y segura, respetando las políticas y procedimientos de seguridad.

En el año 2005 se inició el Green500, que al igual que el TOP500 realiza dos listados al año con un ranking de 500 supercomputadoras. La gran diferencia es que el Green500 lista las 500

supercomputadoras de mayor eficiencia energética del mundo. La técnica utilizada para medir el rendimiento del sistema es la ejecución del benchmark Linpack. (Balladini et al., 2010; Ge et al., 2007; Green500, 2016).

En el año 2007 se fundó The Green Grid, una asociación sin fines de lucro que busca aumentar la eficiencia energética en los centros de datos. Cuenta con la afiliación de las empresas más importantes del mercado. Sus proyectos de investigación, grupos de trabajo y comités a nivel global tienen como finalidad la generación de papers, métricas y herramientas para poder abordar el problema energético (The Green Grid, 2017).

2.4.4.2 Métricas de energía

Como se ha mencionado, el consumo energético es un aspecto fundamental para los diferentes actores involucrados con el sistema. Los diseñadores de hardware y programadores tienen la necesidad y responsabilidad de realizar su trabajo con miras a una eficiente utilización de recursos. Surge la necesidad de contar con técnicas de medición que permitan cuantificar y cualificar a los sistemas desde el punto de vista energético. Por esta razón se han definido métricas, que evalúen nuevos puntos de interés como, por ejemplo: (Balladini *et al.*, 2015; Naiouf M. et al., 2015)

- Caracterización de sistemas desde el punto de vista energético: es necesario conocer el comportamiento energético del sistema para poder realizar comparaciones, y diseñar aplicaciones eficientes en este aspecto. Contar con datos de estudio y análisis contribuye hacia la construcción de modelos de predicción energética y potencia.
- Caracterización de algoritmos complejos desde el punto de vista energético.
- Disponer de técnicas para medir el rendimiento energético de procesadores. Estas estrategias de medición son conocidas como microbenchmarks.
- Modelos de predicción de performance energética: es importante disponer de herramientas que permitan predecir la energía y el rendimiento al ejecutar una aplicación utilizando distintas configuraciones del sistema. De esta forma se podría seleccionar la configuración más adecuada para alcanzar el tiempo de ejecución y eficiencia energética deseados.
- Análisis de esquemas de distribución de procesos entre procesadores, con ajuste dinámico de la frecuencia de clock en función del consumo (overclocking y underclocking).

La potencia es la cantidad de trabajo efectuado por unidad de tiempo. En otras palabras, determina el ritmo al que se consume (o genera) la energía. El watt (o vatio) es la unidad

de potencia y es equivalente a 1 joule por segundo (1 J/s). Dada una computadora con una potencia de consumo de 75 W, si la misma trabaja durante 60 segundos, habrá consumido una energía de 60 x 75 / 3600 = 1,25 Watt/Hora (Montes de Oca, 2012).

La potencia de diseño térmico (TDP - Thermal Design Power) representa la máxima cantidad de potencia que el sistema de refrigeración necesita disipar para mantener al dispositivo debajo de su máxima temperatura. Aunque este valor no es el pico máximo de potencia que se podría disipar, permite tener una idea de la capacidad de disipación de potencia que tienen los procesadores. En algunos procesadores es utilizado para disminuir la frecuencia de reloj o incluso apagar el chip si la temperatura se incrementa por encima de la máxima permitida (Rucci E, 2015; Hennessy & Patterson, 2012; Sopeña *et al.*, 2008).

La métrica más utilizada para evaluar la eficiencia energética es la utilizada por el Green 500. Utiliza el rendimiento por watt. Esto es, la tasa de cómputo, en millones de FLOPS (MFLOPS), que el sistema puede proporcionar por cada W consumido.

$$\boldsymbol{PPW = \frac{Performance}{Potencia}} \qquad \textit{(Fórmula 2.9)}$$

Los valores de Performance y Potencia en la fórmula anterior pueden tener varias definiciones. El Green500 los define como:

- Performance (R_{max}): es el valor máximo de rendimiento alcanzado al ejecutar el benchmark Linpack.
- Potencia ($\bar{P}(R_{max})$): es la media consumida durante la ejecución completa del benchmark Linpack, con un tamaño de problema que proporciona R_{max}.

Por ejemplo, si se utiliza GFLOPS (cantidad de operaciones de punto flotante por segundo en Gigas, donde 1 Giga es igual a 10^9 FLOPS) como unidad de R_{max}, Watt como unidad de $\bar{P}(R_{max})$, y GFLOPS por watt como unidad de "performance por watt", se obtiene:

$$\boldsymbol{GFLOPS\ por\ Watt = \frac{Rmax\ (en\ GFLOPS)}{\bar{P}(Rmax)(en\ Watt)}} \qquad \textit{(Fórmula 2.10)}$$

$\bar{P}(R_{max})$ involucra el consumo de potencia total de todos los nodos de cálculo del sistema, promediados durante la ejecución de Linpack con un tamaño de problema que proporciona el rendimiento máximo R_{max}. Calcular el valor de $\bar{P}(R_{max})$ es una tarea compleja, por esta razón se establecen algunas pautas a tener en cuenta:

- La potencia del sistema se obtiene en base a los nodos de cálculo, excluyendo la potencia consumida por nodos que cumplen otro propósito, por ejemplo, aquellos encargados del almacenamiento.

- Se requiere un tiempo de ejecución Linpack considerable para minimizar la varianza de medición.
- La relación GFLOPS por watt puede cambiar en base a la escala del sistema (cantidad de nodos), el benchmark y el tamaño del problema. Por lo tanto, el $\bar{P}(R_{max})$ debe considerarse para la misma escala y el mismo tamaño del problema Linpack donde el equipo ofrece el rendimiento máximo R_{max}.

Una supercomputadora consiste en un gran número de unidades idénticas (nodos, chasis, racks o armarios, gabinetes, etc.). A menos que la potencia total del sistema esté disponible desde un sistema centralizado de administración de energía, se puede esperar que medir directamente el consumo total de energía de un sistema de supercomputación completo sea una tarea difícil. En algunos casos una alternativa más simple es derivar la potencia total del sistema $\bar{P}(R_{max})$ a partir del consumo de energía de una sola unidad. Si hay N unidades idénticas involucradas en el cálculo del benchmark Linpack y cada unidad consume $\bar{P}_{unit}(R_{max})$, entonces se puede derivar $\bar{P}(R_{max})$ como:

$$\bar{\boldsymbol{P}}(\boldsymbol{R_{max}}) = \boldsymbol{N} \,.\, \bar{\boldsymbol{P}}_{\boldsymbol{unit}}(\boldsymbol{R_{max}}) \qquad \textit{(Fórmula 2.11)}$$

En la ecuación anterior se asumen dos cuestiones:

- Que la carga de trabajo computacional durante la ejecución Linpack se encuentra equilibrada en todas las unidades.
- Que todas las unidades involucradas consumen la misma cantidad de energía para la misma carga de trabajo.

Aunque los dos supuestos planteados suelen ser verdaderos en la mayoría de los casos, una buena práctica es medir varias unidades y utilizar el valor promedio como $\bar{P}_{unit}(R_{max})$ (Ge. et al., 2007).

The Green Grid propone el uso de dos métricas denominadas Power Usage Effectiveness (PUE) y Datacenter Efficiency (DCE), para estimar a corto plazo la eficiencia energética en centros de datos. Estas métricas comparan el total de energía consumida por un centro de datos con la cantidad de energía que realmente llega al equipamiento de IT, lo que permite conocer la cantidad perdida en otros equipos, como por ejemplo los sistemas de refrigeración.

$$\boldsymbol{PUE} = \frac{\boldsymbol{Energía\ del\ equipo\ IT}}{\boldsymbol{Energía\ total\ de\ la\ instalación}} \qquad \textit{(Fórmula 2.12)}$$

Y su recíproco:

$$\boldsymbol{DCE} = \frac{\boldsymbol{Energía\ total\ de\ la\ instalación}}{\boldsymbol{Energía\ del\ equipo\ IT}} \qquad \textit{(Fórmula 2.13)}$$

Donde:

- Energía del equipo IT: es la energía utilizada por aquellos componentes que se utilizan para administrar, procesar, almacenar o enrutar los datos en el centro. Entre ellos se puede mencionar los equipos de computación, almacenamiento y red, monitores, estaciones de trabajo/portátiles utilizadas para monitorear y controlar el centro de datos.
- Energía total de la instalación: es la energía total del centro de datos contemplando componentes de suministro de energía como generadores, UPS y PDUs, componentes utilizados por el sistema de refrigeración, nodos de computación, red y almacenamiento, hasta dispositivos tan sencillos como aquellos utilizados para iluminar el centro de datos.

El PUE y el DCE permiten mejorar la eficiencia operativa, comparar el centro de datos analizado con los centros de datos competitivos, validar si los operadores del centro de datos están mejorando los diseños y procesos, entre otros. Para realizar un análisis a largo plazo, The Green Grid recomienda la utilización de la métrica Datacenter Performance Efficiency (DCPE) y una versión refinada de PUE (The Green Grid, 2007a; The Green Grid, 2007b).

Las métricas de eficiencia para centros de datos se encuentran en constante definición. En los últimos años, ha surgido una gran cantidad de métricas que se encuentran en discusión y permiten analizar distintos escenarios. A modo de ejemplo se puede mencionar: (Kozlowicz, 2014; Global Taskforce, 2014)

- Green Engergy Coefficient / Coeficiente energético verde (GEC): evalúa la cantidad de energía renovable (eólica, solar, geotérmica) utilizada.

$$GEC = \frac{Energía\ verde}{Energía\ total\ de\ la\ instalación} \qquad \textit{(Fórmula 2.14)}$$

- Data Center Energy Productivity / Productividad energética del centro de datos (DCeP): cuantifica el trabajo útil que produce un centro de datos basado en la cantidad de energía que consume.

$$DCeP = \frac{Trabajo\ útil\ producido}{Energía\ total\ consumida\ por\ el\ centro\ de\ datos\ para\ producir\ este\ trabajo} \qquad \textit{(Fórmula 2.15)}$$

- Energy Reuse Factor / Factor de Reutilización de Energía (ERF): esta métrica identifica la porción de energía que se exporta para su reutilización fuera del centro de datos. Un ejemplo sencillo es la utilización del calor generado para calefaccionar las instalaciones.

$$ERF = \frac{Energía\ reutilizada\ fuera\ del\ centro\ de\ datos}{Energía\ total\ del\ centro\ de\ datos} \qquad \textit{(Fórmula 2.16)}$$

2.5 Resumen del Capítulo

En el capítulo 2, se ha detallado una importante cantidad de conceptos relacionados con la concurrencia y el paralelismo de los sistemas informáticos. Se describieron las necesidades humanas que forjaron el camino hacia el paralelismo, así como también las limitaciones sorteadas por los diseñadores de hardware y el proceso de evolución hasta la actualidad, donde se observa que la mayor parte de los productos electrónicos presentan una arquitectura paralela. Los procesadores multicore resultan el eje principal del capítulo; se clasificaron las arquitecturas multicore, se analizó su rendimiento y se puso el foco en dos problemáticas actuales como son el incremento de consumo energético y la temperatura generada por dichos procesadores describiendo las técnicas y tecnologías utilizadas por los diseñadores para atacar estas limitaciones. Se presentan asociaciones e iniciativas surgidas en los últimos años para atender esas problemáticas, las cuales fomentan el desarrollo y utilización de tecnologías Green. Se han definido métricas energéticas utilizadas en la actualidad cuando se analiza un sistema informático. Desde el punto de vista de la programación, se analiza de qué manera resulta afectada por los procesadores multicore y a la vez que surge la necesidad de crear aplicaciones que exploten al máximo todos sus núcleos y que al mismo tiempo resulten eficientes desde el punto de vista energético.

En el próximo capítulo se plantea la necesidad de incorporar los contenidos planteados, concurrencia y paralelismo, al alumno universitario desde las etapas iniciales de su formación académica. Se proponen estrategias y herramientas que permiten abordar los conceptos fundamentales de la concurrencia que, por su naturaleza, son complejos.

CAPITULO 3
TECNOLOGIA INFORMATICA APLICADA A CONCEPTOS BASICOS INFORMATICOS

3.1. Introducción

En el capítulo 2 se han presentado numerosos conceptos informáticos relacionados con la concurrencia y el paralelismo. Se han descripto los procesadores multicore, utilizados en los sistemas de alto rendimiento, computadoras, y en dispositivos tecnológicos como tablets, celulares y consolas de videojuegos, entre otros. La existencia de múltiples núcleos en un mismo chip ha permitido obtener un incremento sustancial en rendimiento, y al mismo tiempo ha modificado el desarrollo de algoritmos para aprovechar al máximo el grado de paralelismo disponible. La generación del calor en un procesador afecta directamente en su funcionamiento y por lo tanto en cómo ejecuta las aplicaciones. Se explicó la problemática del consumo energético en los procesadores y cómo las tendencias actuales llevan hacia un uso eficiente de los recursos informáticos.

La concurrencia, el paralelismo y los conceptos relacionados mencionados, son por naturaleza complejos. Desde el punto de vista de la enseñanza y el aprendizaje de los mismos, se presenta un desafío mayor que cuando se aborda la cuestión de los procesadores de un sólo núcleo donde las aplicaciones se ejecutaban con un solo hilo de control. A pesar de ello, la enseñanza de estos temas en carreras de informática debe incorporarse en etapas tempranas de la formación del alumno. Entre las principales razones se pueden mencionar:

- La necesidad de mantener una currícula vigente y moderna.
- Dado que los problemas del mundo real son esencialmente paralelos, las nuevas arquitecturas si bien son más complejas, también permiten presentar soluciones que se adaptan más eficientemente al modelo de las aplicaciones.
- Las arquitecturas con las que trabajará el alumno durante su etapa de formación académica son paralelas.
- Los algoritmos y aplicaciones actuales deben hacer un uso eficiente de los recursos informáticos, y al mismo tiempo explotar al máximo la arquitectura disponible para obtener soluciones más rápidas.

El avance tecnológico ha proporcionado nuevos escenarios de enseñanza y aprendizaje, así como también herramientas que pueden resultar beneficiosas para abordar diversas temáticas vinculadas con el ámbito educativo. La incorporación de tecnología en este ámbito se dio de

manera paulatina, generando nuevas aéreas de estudio e investigación en tecnología informática aplicada a educación que intentan proveer al educador de herramientas a partir de recursos tecnológicos para mejorar los procesos de enseñanza y aprendizaje. El conjunto de las tecnologías de la información y comunicación (TIC), ha proporcionado una gran cantidad de recursos tecnológicos incorporados a los procesos de enseñanza de forma exitosa. En el capítulo 3, se analizan escenarios educativos que resultan fortalecidos por la utilización de estas herramientas. En particular se focaliza en aquellas herramientas visuales e interactivas utilizadas para la enseñanza de conceptos informáticos, que servirán de referencia para el desarrollo de la herramienta que se propone en esta tesina.

3.2. Dificultades en la enseñanza de conceptos informáticos

Aprender a programar es una tarea compleja para el alumno y un desafío para el docente. En los últimos años se ha añadido a los contenidos tradicionales de programación, nuevos conceptos de concurrencia y paralelismo que complejizan el proceso de enseñanza. La ACM (Association for Computing Machinery) es una asociación fundada en 1960 por las principales sociedades informáticas destinadas a profesionales y científicos. Entre sus objetivos se encuentra la adaptación de los planes estudios en base a la evolución de la ciencia informática. Todos los años elabora un informe en el cual recomienda la creación de nuevas disciplinas y actualización de las ya existentes, que agrupa en cinco sub disciplinas informáticas:

1. Ingeniería Informática
2. Ciencias de la Computación
3. Sistemas de información
4. Tecnología de la información
5. Ingeniería de software

En los últimos planes de estudio se observan recomendaciones para introducir al programa temas como algoritmos paralelos y multihilo, arquitecturas multinúcleo, procesadores DSP y GPUs, aplicaciones beneficiadas por un enfoque multinúcleo, green computing y sustentabilidad (ACM/IEEE-CS, 2013; ACM/IEEE-CS, 2015; ACM, 2017). En general los alumnos muestran dificultades para crear algoritmos, y esta condición se origina en la imposibilidad que presentan cuando deben resolver problemas elementales y, por otro lado, en la reducida capacidad de abstracción (Depetris et al., 2013). En el ámbito universitario y en especial en las carreras informáticas los alumnos de primer año ingresan con algunas de las siguientes dificultades: (Madoz *et al.*, 2005)

- Falta de una adecuada orientación vocacional

El alumno ingresa a una carrera informática sin tener en claro cuáles son los contenidos que ofrece el plan de estudio, ni cuál es el campo de trabajo y aplicaciones laborales en las que podrá ejercer una vez finalizados los estudios.

- Poco entrenamiento en pensar y expresar rigurosamente conceptos

  En general el grado de conocimiento con el que un alumno egresa de la escuela secundaria está determinado por una motivación y esfuerzo personal y no por estándares mínimos que establezca el plan educativo. Normalmente el alumno no ha desarrollado capacidades metacognitivas para poder interpretar, analizar y sintetizar información. La noción de abstracción es insuficiente frente a la clase de problemas que se presentan en el campo informático.

- Dificultad de aprendizaje de los temas básicos

  Las técnicas de estudio utilizadas en la escuela secundaria y la tendencia hacia una cultura general, dificulta la enseñanza de conceptos básicos, concretos y fundamentalmente teóricos, que requieren capacidad de abstracción y la aceptación de reglas rigurosamente especificadas.

- Escasa valoración por el trabajo sistemático

  El modo de trabajo de la escuela secundaria privilegia las soluciones inmediatas (intuición, prueba y error) al desarrollo deductivo o bien inductivo a través de un proceso de elaboración de resultados parciales que conduzcan a conclusiones.

- Gran disparidad de conocimientos y formación previa

  Se aprecia una gran desigualdad, en los alumnos ingresantes a una carrera universitaria, en cuanto a los contenidos, metodologías de estudio y capacidades para interpretar, analizar y resolver problemas.

En general, estas son algunas de las problemáticas que presentan los alumnos al ingresar a una carrera universitaria. En particular, para las carreras informáticas es de vital importancia, para el alumno, la interpretación, el poder de análisis, la facultad de pensar de forma lógica, y en especial de la capacidad de abstracción. El proceso de aprendizaje en los alumnos suele ser más eficiente cuando avanza de lo concreto a lo abstracto. Si bien un adulto debe tener la capacidad de entender conceptos abstractos, pensar de forma lógica y generalizar, la posibilidad de poder presentar contenidos a través de los sentidos (visual, táctil, auditivo), beneficia de forma notoria el proceso de enseñanza.

Las prácticas educativas intentan atender las problemáticas mencionadas previamente y han sido modificadas con el objeto de mejorar su calidad de enseñanza. El concepto de mediación subyace a la práctica educativa de varias formas: entre los contenidos y el alumno, entre la metodología y los contenidos, entre el alumno y el docente o entre los grupos de alumnos. Estas

relaciones han cambiado en base a las manifestaciones tecnológicas, desde las más sencillas hasta las más sofisticadas, ya que desde hace un tiempo atrás la tecnología es utilizada para enseñar y aprender (Zangara, 2014). La tecnología ha proporcionado una gran cantidad de herramientas que permiten presentar al alumno contenidos a través de los sentidos (visual, táctil, auditivo), mediante el uso de ejemplos concretos. En las próximas secciones se describe de qué manera se incorporan recursos tecnológicos a los procesos educativos, las herramientas que se encuentran disponibles y los beneficios obtenidos al utilizarlas, focalizando especialmente en aquellas herramientas visuales e interactivas que facilitan el aprendizaje de conceptos informáticos. La incorporación e integración de recursos tecnológicos en el ámbito de la educación ha sido un proceso lento que es descripto por la Tecnología Educativa.

3.3. La tecnología en la educación

La tecnología educativa es una disciplina originada en Estados Unidos en la década de 1950. La necesidad de formar una gran cantidad de soldados (en destrezas específicas según las tareas a desempeñar en la organización militar) para la II Guerra mundial tuvo como resultado procesos eficaces de enseñanza que utilizaban medios y recursos técnicos. Esta primera iniciativa fue evolucionando a lo largo del tiempo y las distintas asociaciones profesionales y académicas han formado y trasformado el concepto de tecnología educativa dando lugar a diferentes enfoques como son la enseñanza audiovisual, enseñanza programada, tecnología instruccional, diseño curricular, tecnología crítica de la enseñanza, entre otros (Moreira, 2009; Cabero, 2007).

La tecnología educativa intenta proveer al educador de herramientas a partir de la utilización de recursos tecnológicos con el objetivo de mejorar los procesos de enseñanza y aprendizaje. Entre sus principales ventajas se encuentran el acceso a la información completa (de diferentes fuentes) de forma inmediata lo que incrementa el nivel académico del alumno, la posibilidad de aprender de forma interactiva y a distancia (sin limitación de tiempo y lugar), la constante evolución de internet y sus beneficios como son el correo electrónico, conferencias, redes sociales, entre otros. Como desventajas se puede mencionar la necesidad de contar con los recursos físicos como computadoras y acceso a internet, la necesidad de saber utilizar estas herramientas y la posibilidad de no obtener los resultados esperados al realizar una mala utilización de los recursos tecnológicos.

Según Area (Area, 2004) la tecnología educativa en la actualidad está marcada por los siguientes ejes conceptuales:

- Concentra una gran cantidad de conocimiento pedagógico de diferentes aéreas como la cultura, la educación, la tecnología, entre otras; y recibe aportes desde otras disciplinas de las ciencias sociales.
- Es una disciplina que investiga cómo sacar provecho de los avances tecnológicos en los procesos de enseñanza que serán utilizados en diversos contextos educativos.
- La naturaleza del conocimiento de la Tecnología Educativa es afectada a partir de los intereses y valores que subyacen a los proyectos sociales y políticos en los que se inserta la elaboración, uso y evaluación de la tecnología.
- En la actualidad concibe a los medios y tecnologías de la información y comunicación como objetos o herramientas culturales que los individuos y grupos sociales reinterpretan y utilizan en función de sus propios esquemas o parámetros culturales.
- Depende de cómo se relacionan los individuos con la tecnología en diferentes contextos sociales, culturales, ideológicos, entre otros.
- Los métodos de estudio e investigación de la Tecnología Educativa combinan aproximaciones cuantitativas con cualitativas en función de los objetivos y naturaleza de la realidad estudiada. Generalmente adoptan una posición intermedia o indefinida, sin oponerse a ninguna de las posiciones posibles.

Hoy en día se puede definir la tecnología educativa como el resultado de aplicar diferentes concepciones y teorías educativas para la resolución de un amplio espectro de problemas y situaciones referidos a la enseñanza y el aprendizaje, apoyadas en las Tecnologías de la Información y Comunicación (Olguín, 2012).

3.3.1. Tecnologías de la Información y Comunicación (TIC)

La Tecnología Educativa como disciplina tuvo un gran crecimiento y aceptación con la aparición de las tecnologías de la información y comunicación (TIC).

Julio Cabero, referente en Tecnología Educativa y nuevas tecnologías aplicadas a la educación define las TIC como: “En líneas generales podríamos decir que las nuevas tecnologías de la información y comunicación son las que giran en torno a tres medios básicos: la informática, la microelectrónica y las telecomunicaciones; pero giran, no sólo de forma aislada, sino lo que es más significativo de manera interactiva e interconexionadas, lo que permite conseguir nuevas realidades comunicativas” (Cabero, 1998).

Las TIC son el conjunto de tecnologías producidas en los ámbitos de la informática y la comunicación, que posibilitan el acceso, producción, manipulación y comunicación de información que se presenta a través de distintos medios (texto, imagen, sonido, entre otros). Su

origen es posterior a la Tecnología Educativa, la cual tuvo un crecimiento exponencial gracias a dos elementos tecnológicos: la computadora e internet. Las computadoras permiten trabajar con aplicaciones informáticas como por ejemplo editores de texto, imágenes, aplicaciones multimedia, entre otros. El uso de internet creció rápidamente convirtiéndose en un sistema global de comunicación que permite acceder a información en forma inmediata donde la distancia física no es una limitación.

Las TIC colaboran en el ámbito educativo. A continuación, se describen algunas de sus principales características que favorecen el proceso de aprendizaje: (Belloch, 2012)

- Interactividad

  La interactividad hace referencia al proceso de comunicación entre humanos y computadoras, esto es la relación de participación que se produce entre los usuarios y los sistemas informáticos. Aunque la relación personal presencial entre un alumno y su profesor es desde el punto de vista comunicacional más beneficiosa, las TIC han reducido esta brecha por ejemplo mediante la utilización de conferencias audiovisuales en vivo. En aquellos casos donde la interacción presencial resulta imposible o poco frecuente por limitaciones como la distancia o disponibilidad horaria, las TIC lo hacen posible, tanto de forma sincrónica como asincrónica. La interactividad permite adaptar los recursos educativos en base a las necesidades y características del usuario final.
- Interconexión

  Esta característica hace referencia a la posibilidad de crear nuevas tecnologías al combinar tecnologías existentes. Un ejemplo es la telemática que surge al conectar la informática con las tecnologías de comunicación y provee nuevos recursos como son el correo electrónico, la transferencia de ficheros (ftp) y la web, entre otros.
- Información multimedia de alta calidad

  Las TIC trabajan con todo tipo de información: textual, imagen y sonido, las cuales se encuentran en formato digital de gran calidad.
- Digitalización

  Es el proceso que permite convertir la información que se encuentra en diferentes formatos (audio, texto, imágenes, etc.) en formato digital. El objetivo final es que los usuarios puedan acceder a esta información ubicada en dispositivos electrónicos distribuidos geográficamente, pero interconectados por redes de comunicación.
- Inmaterialidad

  La materia prima de las TIC es la información y su digitalización posibilita el acceso a la misma es inmaterial y transparente para los usuarios.
- Instantaneidad

Posibilita el acceso inmediato a la información a través de las redes de comunicación y su integración con la informática.

- Innovación

  Su carácter innovador y creativo, genera nuevas formas de comunicación y cambios en todos los ámbitos sociales.

Las TIC permiten innovar en el ámbito educativo y replantear los procesos de enseñanza y aprendizaje. En los últimos años se han ido integrando a los procesos de enseñanza, pero para ello es necesario analizar y evaluar los recursos tecnológicos de acuerdo al uso que se haga de ellos, crear estrategias educativas para integrar estos recursos en diferentes ambientes de enseñanza como cursos presenciales o a distancia, individuales o grupales, entre otros. La utilización de las TIC implica crear material digital/multimedia y cursos que hagan uso de ellas y al mismo tiempo requiere que los profesionales estén capacitados para trabajar con estos recursos.

En la Tabla 3.1 se puede observar algunas de las ventajas y desventajas del uso de las TIC: (Gómez, 2015; Semenov, 2005; Calzadilla, 2002)

Ventajas	**Desventajas**
Favorecen el aprendizaje colaborativo.	Se genera una dependencia a la disponibilidad y correcto funcionamiento de los sistemas informáticos que son utilizados.
Incrementan la creatividad y proponen continuas actividades intelectuales.	Se requiere tener conocimientos para utilizar las tecnologías utilizadas.
Permiten el acceso a gran volumen de información y los usuarios desarrollan habilidades de búsqueda y selección de información.	La gran cantidad de información y el libre acceso a tecnologías como internet puede ocasionar que el alumno trabaje con fuentes poco confiables, y sufra distracciones o pérdidas de tiempo.
Proveen mecanismos de aprendizaje flexibles y personalizados, que suelen ser de gran ayuda para enseñar a alumnos con capacidades diferentes.	
Los alumnos frecuentemente se sienten más atraídos por el tipo de contenidos que proveen las TIC, disminuyendo el tiempo de aprendizaje.	

Tabla 3.1: TIC - Ventajas y Desventajas.

3.3.2. Estrategias y escenarios de aprendizaje mediados por tecnología

Se ha definido el concepto de TIC (tecnologías de la información y comunicación) indicando cuáles son sus principales características y que beneficios brindan al ser incorporadas en el ámbito educativo. Los escenarios de aprendizaje representan una parte dinámica en el proceso de enseñanza, de donde se desprenden líneas de investigación para diseñar nuevas alternativas que contribuyan en la educación del alumno adecuándose a las necesidades actuales y sacando provecho del avance tecnológico. La introducción e integración de TIC en los escenarios educativos ha generado un nuevo campo de investigación y debate del que se desprenden una gran cantidad de temas que pueden ser resumidos en 3 ejes: (Sanz *et al.*, 2014a)

- Trabajo colaborativo mediado por TIC.

  El trabajo colaborativo requiere de un grupo de personas que trabajan en conjunto de forma cooperante para lograr una meta. Cuando este trabajo se realiza dentro del ámbito educativo y mediado por TIC, se hace referencia al conjunto de técnicas y estrategias de enseñanza que son apoyadas por la tecnología. Estas estrategias permiten que cada integrante del grupo desarrolle habilidades (de aprendizaje y de desarrollo personal y social), y sea responsable de su propio aprendizaje, pero también del resto de los miembros del grupo (Gros, 2006). Las TIC proporcionan una amplia gama de herramientas que permiten la creación de espacios de trabajo que acompañan y enriquecen la colaboración e interacción de los participantes. Se cuenta con herramientas sencillas como son los blogs y las wikis, hasta otras más robustas como los entornos virtuales de enseñanza y aprendizaje (Sanz *et al.*, 2008).

- Diseño, producción y evaluación de Materiales educativos digitales. Objetos de aprendizaje y sus Repositorios. Simuladores. Laboratorios Remotos y Virtuales.

  La generación de materiales educativos digitales es una tarea imprescindible para poder enseñar en escenarios mediados por la tecnología. Se cuenta con la posibilidad de utilizar diversos medios para poder presentar y estructurar la información, como son la ilustración, el audio, el video, la animación, hipertextos, tecnología web, entre otros. Al crear materiales digitales se debe tener en cuenta el contenido y a quien será destinado el material, pero también cuál será el medio de uso (portales, blogs, repositorios, entornos virtuales, entre otros). El docente no sólo debe conocer y saber utilizar las tecnologías de enseñanza actuales, sino también cómo diseñar nuevos contenidos (Russo *et. al*, 2014).

  Los objetos de aprendizaje pertenecen a la categoría de materiales educativos, su finalidad es educativa y su proceso de diseño espera que un OA cuente con ciertas

características como la reutilización, la interoperabilidad, la durabilidad, el acceso desde diferentes repositorios, entre otras. Los OA son un tipo de material educativo particular que ha tomado relevancia en los últimos años. Se encuentran alojados en repositorios que posibilitan el acceso a docentes y alumnos para utilizarlos como herramientas de enseñanza y aprendizaje. Es deseable que el docente incursione no sólo en el uso de los OA, sino también en el diseño y producción de los mismos (Sanz *et al.*, 2014b).

Los laboratorios permiten que los alumnos tengan un contacto más cercano y práctico con los contenidos estudiados. Las TIC proveen una alternativa conocida como laboratorios de aprendizaje virtual o remoto, que mediante el uso de simulaciones permite realizar trabajos prácticos y experimentos. Los laboratorios virtuales no intentan reemplazar a los laboratorios físicos, sino proveer otra opción diferente sin limitaciones de espacio y tiempo, que en algunos casos disminuye los riesgos de manipular material peligroso o reducir notablemente los costos económicos (Sanz *et al.*, 2014a).

- Entornos virtuales de Enseñanza y Aprendizaje (EVEA). Campus virtuales. Entornos inmersivos. Sistemas multimediales e hipermediales. Redes sociales.
  A finales de los años 80 empezaron a utilizarse diferentes aplicaciones que resultaron en lo que hoy se conoce como Entornos Virtuales de Enseñanza y Aprendizaje (Salinas, 2012), y desde entonces el desarrollo e inclusión de nuevos entornos de aprendizaje afectó tanto el rol del alumno como el del profesor. Surge la necesidad de contar con un docente especializado en un nuevo perfil tecnológico, que cuenta con los conocimientos necesarios para utilizar y administrar el entorno, reconocer cuáles son sus ventajas y desventajas, y así poder hacer un uso eficiente del mismo y que pueda explotar toda su funcionalidad y aproveche los puntos más fuertes de la herramienta. Se emplea un nuevo modelo pedagógico que concibe a las instituciones como instituciones de gestión de conocimiento (Salinas, 2012), donde el alumno es el trabajador y el docente es quien diseña el trabajo del estudiante. Este paradigma persigue la formación de un alumno que se encuentre motivado y pueda ser autodirigido, empleando sus propios recursos para aprender (buscar y organizar la información, relacionarse, plantear preguntas, entre otros) (Sanz *et al.*, 2015).

  Los entornos virtuales de enseñanza y aprendizaje se han extendido en un nuevo concepto de mayor magnitud conocido como campus virtual, que engloba otros servicios y herramientas de una institución educativa. Se facilita la comunicación entre profesores y alumnos a partir de foros de contacto, publicación de material de estudio, carteleras virtuales; y se ofrecen servicios administrativos que son útiles para el alumno,

como la consulta de resultados de exámenes, asistencias y horarios, trámites académicos, entre otros.

Los entornos virtuales inmersivos, (ambientes virtuales inmersivos, AVIs o metaversos) son entornos tridimensionales generados por computadora que recrean escenarios reales o imaginarios. Aquí cada usuario tiene una representación virtual (conocida como avatar) con la que puede interactuar con el mundo virtual y con otros usuarios del entorno. Los AVIs son utilizados en el ámbito educativo para ofrecer al alumno mayor interactividad que las plataformas bidimensionales, y una oportunidad de adquirir conocimientos en línea (Altube *et al.*, 2016).

Los sistemas multimediales e hipermediales permiten integrar múltiples medios estimulando el interés del alumno, fomentando la actividad creativa, manipulando información no secuencial ni estructurada y sacando provecho a las capacidades que cuentan los estudiantes para trabajar con este tipo de herramientas. Los sistemas multimediales se refieren al uso de medios de expresión físicos o digitales (desde texto e imágenes, hasta animación, sonido y video, entre otros) para presentar e integrar información. Entre las principales ventajas se puede mencionar una visualización atractiva y adaptada (como por ejemplo el manejo de múltiples idiomas) de información disponible en diferentes plataformas que promueve una participación activa (Beneforti & Ainchil, 2000).

Las redes sociales pueden ser una herramienta utilizada en el ámbito educativo para favorecer el aprendizaje. Brindan la posibilidad de compartir documentos, enlaces web, videos; tener una comunicación eficaz entre profesor-alumnos o alumnos-alumnos mediante el uso de foros, chats, llamadas o videoconferencias. En muchos casos las redes sociales se utilizan como un complemento dentro de un curso de modalidad presencial, con el objetivo de incrementar la interacción del grupo en donde todos pueden generar contenido, debatir y aprender al mismo tiempo (Gomez & Redondo, 2011).

3.4. Enfoques y herramientas en la enseñanza de un curso de computación

Hasta aquí se han presentado algunas estrategias y escenarios de aprendizaje mediados por tecnología informática, que brindan diferentes alternativas para llevar adelante los procesos de enseñanza y aprendizaje. En esta sección se evalúan distintos enfoques educativos para la enseñanza de contenidos informáticos, en especial aquellos relacionados con la programación y se analiza la utilización de herramientas tecnológicas para abordar estos contenidos.

En sus inicios, los cursos de programación pusieron el foco en la enseñanza de lenguajes por encima de las técnicas y buenas prácticas de programación. Con el paso del tiempo los contenidos curriculares fueron adaptándose anteponiendo el análisis y el diseño de algoritmos al lenguaje de programación utilizado. Se introdujeron temas relacionados con la modelización y la abstracción de problemas del mundo real, el diseño top down, los refinamientos sucesivos, tipos de datos abstractos, diseño de patrones, invariantes algorítmicos, corrección y eficiencia de algoritmos (De Giusti *et al.*, 2003). Como se mencionó previamente resulta un desafío la enseñanza de estos temas a los alumnos en las etapas iniciales de las carreras informáticas, y a esto se suma la necesidad de incorporar nuevos contenidos que han surgido a partir de los avances tecnológicos, como por ejemplo algoritmos paralelos y multihilo, arquitecturas multinucleo, procesadores DSP y GPUs, aplicaciones beneficiadas por un enfoque multinucleo, green computing y sustentabilidad, entre otros. La tarea del docente para la enseñanza de estos temas es compleja y es necesario buscar distintas estrategias para solucionar los inconvenientes que se presentan, contemplando el uso de recursos atractivos para los alumnos. Aquí las tecnologías de la comunicación e información pueden ser adoptadas e integradas para beneficiar los procesos educativos. A su vez, la informática y los recursos tecnológicos como los sistemas distribuidos, las redes, internet y la multimedia, brindan otras alternativas y herramientas que pueden beneficiar los procesos de aprendizaje. La tecnología informática incrementa la potencialidad e el interés del alumno permitiendo adecuarse a las necesidades individuales (Sanz *et al.*, 2014a; Sanz *et al.*, 2015).

En base a las características de las carreras informáticas y del perfil de sus alumnos parece adecuado incorporar, al proceso educativo, la utilización de materiales hipermediales, materiales multimediales, herramientas visuales y software educativo, entre otros, con el objeto de facilitar el aprendizaje de estos contenidos cada vez más complejos (De Giusti *et al.*, 2003). Es importante destacar que en poco tiempo, todo ingresante universitario habrá interactuado desde su infancia con una gran variedad de dispositivos electrónicos y aplicaciones. En especial, los ingresantes a carreras informáticas poseen experiencia y demuestran interés para trabajar con entornos gráficos e interactivos, y la ACM (Association for Computing Machinery) recomienda utilizar este tipo de herramientas para la enseñanza de conceptos, particularmente en cursos introductorios de programación (ACM/IEEE-CS, 2004; ACM/IEEE-CS, 2008; ACM/IEEE-CS, 2013). El objetivo principal es facilitar la comunicación entre el profesor y el estudiante para favorecer, a través de la intuición y el razonamiento, un acercamiento comprensivo de las ideas a través de los sentidos. Los contenidos curriculares de las asignaturas informáticas, en general, contienen una diversidad de temas que obligan al alumno a realizar una abstracción para lograr comprenderlos, y la posibilidad de visualizarlos de forma aplicada y gráfica en un entorno adecuado puede beneficiar el aprendizaje de los mismos.

3.4.1. Herramientas Visuales en un curso de programación

Históricamente la utilización de un paradigma imperativo y la programación estructurada ha sido la opción elegida para introducir al alumno en la programación. Con el tiempo el paradigma orientado a objetos fue tomando mayor relevancia y se convirtió en un enfoque más innovador que introduce una gran cantidad de conceptos que pertenecen a la mayoría de los lenguajes de programación vigentes en la actualidad. En ambos casos los lenguajes utilizados son textuales, esto significa que los algoritmos serán expresados mediante cadenas de texto con una sintaxis determinada según el caso.

En cambio, la programación visual comprende el uso de expresiones visuales (tales como gráficos, dibujos, animaciones o íconos) en el proceso de programación. Permite desarrollar aplicaciones desde un entorno amigable y de fácil usabilidad para el programador. Las expresiones visuales suelen ser traducidas a un lenguaje de programación textual, o pueden ser empleadas para formar la sintaxis de nuevos lenguajes de programación visual conduciendo a nuevos paradigmas tales como la programación por demostración, o pueden ser utilizas en la representación gráfica del comportamiento o estructura de un programa (Hernández, 2013; McIntyre, 1992). La programación visual aprovecha el hecho de que los humanos procesan mejor y más rápido las imágenes que el texto, permitiendo obtener más información, en menor tiempo. Entre los argumentos más relevantes se puede mencionar los siguientes:

- El texto es unidimensional y requiere un acceso secuencial. Las imágenes son multidimensionales y permiten un acceso no secuencial.
- Las imágenes proveen una visión global o detallada dependiendo del caso, con un lenguaje enriquecido mediante el uso de colores, formas, tamaños, entre otros.
- El sistema sensorial humano procesa las imágenes en tiempo real.

Estas ventajas pueden ser aprovechadas para diseñar herramientas visuales que resulten útiles en el ámbito educativo. Cuando se trabaja en un curso de programación es deseable que los alumnos puedan escribir sus algoritmos y visualizar gráficamente la ejecución de los mismos para poder validar si sus soluciones son correctas o no. Se espera también que el alumno pueda comparar distintas soluciones a un mismo problema, que detecte los errores cuando se le plantea un algoritmo determinado y en consecuencia pueda visualizar el comportamiento no deseado que se produce, y que pueda realizar cambios sobre una solución dada para analizar su comportamiento. En la Tabla 3.2 se mencionan algunas de las herramientas utilizadas para la enseñanza de la programación en el ámbito educativo:

Herramienta	Objetivo educativo	Descripción del entorno	Características principales
Cubik (García *et al.*, 1996)	Entorno desarrollado por la Universidad Nacional del Sur para enseñar a programar mediante la implementación de algoritmos estructurados.	La herramienta cuenta con una ventana para la edición de algoritmos utilizando un pseudocódigo sencillo para el alumno. Este algoritmo luego es traducido al lenguaje Pascal y puede accederse y modificarse desde una segunda ventana de edición de textos.	Posee una interfaz amigable, gráfica y compatible con los sistemas operativos actuales. Realiza un análisis léxico, sintáctico y semántico, que facilita la tarea de programación del alumno y evita incurrir en errores propios del lenguaje.
Karel el Robot (Becker Weber, 2001; Pattis, 1981)	El lenguaje Karel fue creado por Richard Pattis y utilizado en sus clases para que los estudiantes aprendieran a pensar de forma sistematizada y eficiente.	Incorpora el uso de robots que ejecutan acciones, en base a las instrucciones que provee el lenguaje. El robot se encuentra en un mundo formado por avenidas y calles, donde podrá trasladarse, recoger objetos y llevarlos hacia otro lugar. La ejecución es visualizada de forma gráfica permitiendo al alumno escribir y ver la ejecución de sus propios algoritmos. Esta herramienta ha sido tomada en cuenta para generar nuevas herramientas que tengan este mismo enfoque.	Posee instrucciones sencillas y bien estructuradas, permitiendo escribir algoritmos sin necesidad de conocer algún otro lenguaje de programación. Es una herramienta que promueve la creatividad y la lógica de una manera ordenada, sentando bases sólidas para el aprendizaje de la programación.
Robótica basada en Ada (Fagin, 2001; Fagin 2000)	Es una herramienta creada por Barry Fagin para la enseñanza de conceptos básicos de computación.	Se utiliza robótica basada en Ada y mediante la presentación de una amplia variedad de ejemplos e implementación de algoritmos, el alumno puede aprender conceptos básicos de programación como flujos de control, variables, constantes, modularización, estructuras de datos, estructuras de control, entre otros.	La posibilidad de visualizar el comportamiento del robot en base al algoritmo implementado permite al alumno verificar de forma real y concreta el resultado de su código. Al mismo tiempo incrementa su interés incorporando un factor de diversión al aprendizaje.
Visual Da Vinci (De Giusti, 2003)	Entorno visual creado por la Facultad de Informática de la UNLP y utilizado para introducir conceptos de programación como estructuras de control, tipos de datos simples, y modularización entre otros.	Al igual que Karel el Robot, aquí se cuenta con un robot que se ubica en una ciudad compuesta por avenidas y calles. Dicho robot entiende instrucciones que le permiten trasladarse, tomar y depositar flores/papeles, entre otras. El alumno puede escribir sus algoritmos en un editor utilizando un lenguaje sencillo, y puede visualizar gráficamente el comportamiento del robot al ejecutar el código implementado.	Utiliza un lenguaje sencillo que permite introducir al alumno en la programación para luego avanzar a un lenguaje de programación más completo. El alumno se siente motivado al utilizar una herramienta interactiva donde puede visualizar la ejecución de su algoritmo.

Tabla 3.2: Herramientas visuales para la enseñanza de la programación.

Se han descripto algunas herramientas visuales en las cuales los alumnos pueden escribir soluciones para resolver problemas y verificar su funcionamiento mediante la visualización del algoritmo en ejecución. En particular, aquellas herramientas que utilizan robots resultan adecuadas en cursos de programación inicial y en casos donde es necesario enseñar conceptos en los que el alumno deben realizar un proceso de abstracción. En su mayoría, estas herramientas definen un lenguaje de programación que trabaja con un conjunto de instrucciones sencillas y que resultan sencillas de entender para el alumno, evitando así la necesidad de enseñar un lenguaje de programación de mayor complejidad o que requiera que el alumno posea conocimientos previos. La simulación gráfica de la solución implementada a un problema dado ofrece al alumno un mecanismo de autocorrección (verificación) al comparar el resultado esperado y el resultado de su ejecución. Los alumnos podrán identificar más fácilmente en qué lugar del código se encuentra el error simplemente observando la animación, aprovechando el hecho de que el cerebro humano tiene una mejor performance para procesar una información visual. Es necesario aclarar a los alumnos que aún cuando la simulación gráfica resuelve el problema y el resultado final es el esperado, no se puede afirmar que el algoritmo es correcto. Es necesario contar con una segunda validación que realiza el profesor en la que se controlan otros aspectos como, por ejemplo, la utilización de buenas prácticas en la programación, que el algoritmo tenga un grado de eficiencia aceptable, que los tipos de datos y estructuras sean las adecuadas, entre otros.

El entorno Visual Da Vinci mencionado previamente, es un caso de éxito utilizado desde hace años en la Facultad de Informática de la UNLP. En los próximos capítulos se realizará una descripción en detalle del mismo, detallando su evolución a través del tiempo. Desde el punto de vista pedagógico y en base a la experiencia de trabajo se han detectado las siguientes ventajas: (De Giusti, 2003)

- ✓ Los estudiantes se sienten atraídos a resolver problemas concretos que tienen una relación directa con el aprendizaje de conceptos teóricos.
- ✓ Simplifica el pasaje hacia un lenguaje de programación más completo, una vez que el alumno ya ha adquirido los conocimientos básicos de programación dentro de un entorno limitado, y en consecuencia más simple.
- ✓ Permite enseñar de forma natural conceptos de modularización y otros relacionados como: el manejo de datos locales y globales, pasaje de parámetros, entre otros.
- ✓ El alumno generalmente quiere acceder de forma prematura a herramientas profesionales de mayor complejidad descuidando el objetivo central que es la adquisición y maduración de conceptos básicos de programación, prevaleciendo la utilización de buenas prácticas. En este sentido Visual Da Vinci ofrece límites para que

el estudiante no se disperse incursionando en contenidos no contemplados durante el curso.

La herramienta tiene algunas desventajas: (De Giusti, 2003)

- ✓ El alumno suele ser tentado a resolver problemas mediante "prueba y error", ejecutando sucesivas veces y modificando el algoritmo hasta obtener el resultado esperado. Lo deseable es que los alumnos comprendan el enunciado, analicen que deben hacer y cómo resolverlo, sin recurrir a la verificación constante de soluciones inmediatas y carentes de análisis.
- ✓ En aquellos casos donde la clase de problemas que debe resolver el alumno son demasiado simples y no requieren demasiada abstracción, el entorno puede agregar una complejidad adicional sin obtener un beneficio que justifique su uso.
- ✓ El entorno tiene algunos defectos propios de un desarrollo de software. Una de las principales críticas es que los mensajes de error no explican claramente cuál es el problema. El alumno puede tener dificultades para corregir errores sintácticos como la indentación o respetar las mayúsculas/minúsculas de palabras claves.

3.5. Resumen del Capítulo

En este capítulo se han analizado temas vinculados a diversos enfoques y herramientas tecnológicas utilizadas en el ámbito educativo, particularmente para la enseñanza de la programación. Se ha planteado la necesidad de incorporar a las currículas de carreras informáticas los conceptos de concurrencia y paralelismo tratados en el capítulo 2 y descripto las problemáticas existentes en los procesos de enseñanza y aprendizaje, en parte por la complejidad de la temática, pero también por algunas problemáticas que presenta el alumno de informática al ingresar a la universidad. Intentando resolver estas cuestiones se investiga la utilización de la tecnología en el ámbito educativo, detallando una amplia gama de alternativas que ofrecen distintas prestaciones en base a las necesidades, objetivos y limitaciones de los docentes y alumnos. Se ha expuesto algunas herramientas tecnológicas utilizadas para la enseñanza de conceptos informáticos, y se extiende en aquellas que poseen características visuales y un alto grado de interacción con los alumnos. Se presentó una breve descripción de estos entornos interactivos como son Cubik, Karel el Robot, Robótica basada en Ada y Visual Da Vinci. Este último es un entorno que ha evolucionado a lo largo del tiempo y permite enseñar conceptos básicos de concurrencia. A continuación, en el capítulo 4, se detalla la herramienta Visual Da Vinci y su evolución a CMRE (Concurrent Multi Robot Environment), explicando su estado actual para finalmente definir e implementar un prototipo denominado

ECMRE (Extended Concurrent Multi Robot Environment) que incorpora los conceptos de concurrencia y paralelismo presentados en el capítulo 2.

CAPITULO 4
CMRE (CONCURRENT MULTI ROBOT ENVIRONMENT)

4.1 Introducción

En el capítulo 3 se ha planteado la necesidad de incorporar a las currículas de carreras informáticas los conceptos de concurrencia y paralelismo tratados en el capítulo 2. Se realizó una descripción del desafío que representa para el docente la enseñanza de estos temas y como la tecnología puede colaborar en el proceso de enseñanza-aprendizaje mediante la incorporación de herramientas tecnológicas. Se presentó una amplia gama de alternativas que se ajustan a diferentes escenarios educativos, y en particular se focalizó en aquellas herramientas utilizadas para enseñar conceptos informáticos. Sobre este último grupo, se enumeraron algunos entornos de aprendizaje visuales e interactivos como son Cubik, Karel el Robot, Robótica basada en Ada, Visual DaVinci y su evolución CMRE.

En este capítulo, se realiza una descripción del entorno CMRE (Concurrent Multi Robot Environment) en su versión actual, la cual permite enseñar conceptos básicos de concurrencia y paralelismo en las asignaturas de los primeros años de las carreras de la Facultad de Informática de la UNLP. El entorno CMRE es extendido en el capítulo 5 para conformar el ECMRE (Extended Concurrent Multi Robot Environment), permitiendo enseñar al alumno los conceptos tratados en el capítulo 2. CMRE surge a partir del entorno Visual DaVinci que se ha utilizado en la introducción a la programación en varias Universidades, y por esta razón es necesario comenzar describiendo el entorno Visual DaVinci antes de presentar a CMRE en su versión actual.

4.2 Visual DaVinci

Visual DaVinci es un lenguaje de programación visual desarrollado para la enseñanza y el aprendizaje de programación a nivel inicial. Los conceptos proporcionados responden a investigaciones realizadas en el III-LIDI (Instituto de Investigación en Informática LIDI) donde se define una maquina abstracta, representada por un robot inicialmente llamado Lubo-1 que se traslada dentro de una ciudad y mediante un lenguaje de programación asociado es posible definir su comportamiento. El objetivo de este lenguaje es introducir al alumno a la programación estructurada mediante la implementación de algoritmos que son escritos como una secuencia de instrucciones, o a través un diagrama visual dentro de un ambiente amigable (Champredonde *et al.*, 1999).

La versión inicial del Visual DaVinci fue influenciada por otros lenguajes como Ada, Occam, Pascal y C. La implementación del ambiente se realizó utilizando Delphi ya que era un lenguaje que facilitaba la creación de la interfaz, es orientado a objetos y sus tiempos de compilación y ejecución resultaron eficientes. Al ser pensado con fines educativos se decidió crear un lenguaje visual que permita programar al robot. En sus inicios, el desarrollo de los algoritmos se podía implementar a partir de la confección de un diagrama visual que luego era convertido automáticamente a código textual. En las versiones posteriores la alternativa del diagrama visual fue descartada, priorizando que el alumno escriba sus algoritmos de forma textual en un editor de código (Champredonde R. *et al.*, 1999; Champredonde & De Giusti, 1996).

4.2.1 El Robot

El robot puede recorrer una ciudad (ver Figura 4.1) representada como un cuadrado de 100 calles (horizontales) y 100 avenidas (verticales) mediante la ejecución de una instrucción que lo lleva de una esquina hacia otra de acuerdo a la dirección actual. Cada esquina es una intersección entre una avenida y una calle, en la que es posible encontrar flores y papeles.

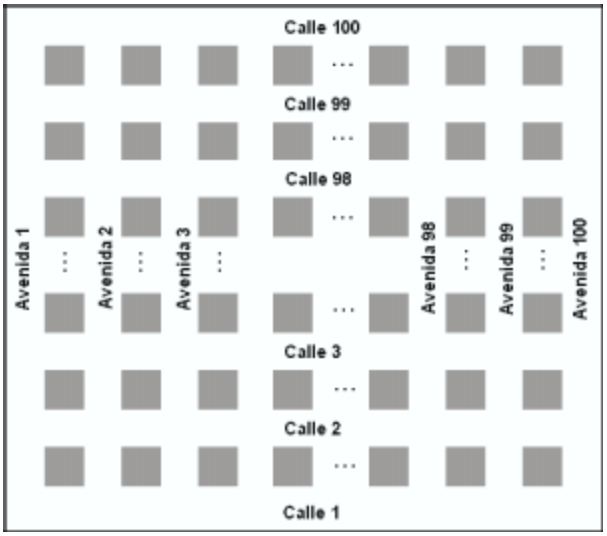


Figura 4.1: Ciudad del robot.

El conjunto de capacidades que posee el robot se puede resumir en el siguiente listado: (Champredonde *et al.*, 1996; Champredonde et al., 1999; III-LIDI, 2017)

- Se mueve.
- Gira hacia la derecha para modificar su orientación. En cada giro se produce una rotación de 90 grados en el sentido de las agujas del reloj.
- Posee la habilidad de reconocer dos tipos de objetos que son flores y papeles. Estos objetos se encuentran distribuidos en las esquinas de la ciudad.

- Cuenta con dos bolsas en las que puede transportar flores y papeles. Las bolsas tienen una capacidad ilimitada. El robot puede recoger y/o depositar cualquiera de los dos tipos de objetos en una esquina, pero de a uno a la vez. Antes de intentar tomar una flor o papel en una esquina en particular, es necesario verificar la existencia del elemento en cuestión, ya en caso de no existir se produce un error en ejecución. Del mismo modo antes de depositar una flor o un papel el robot debe verificar que su bolsa no se encuentre vacía, en caso contrario se produce un error en ejecución.
- Puede realizar cálculos simples.
- Puede informar los resultados obtenidos.

Se ha definido que acciones puede realizar el robot dentro de la ciudad. Estas acciones se corresponden con instrucciones propias del lenguaje que el robot sabe interpretar tanto sintáctica como semánticamente. En la Tabla 4.1 se definen las instrucciones concretas que permiten al robot desplazarse, tomar, depositar, evaluar algunas condiciones sencillas y visualizar información.

Sintaxis	**Semántica**
iniciar	Instrucción primitiva que posiciona al robot en la esquina (1,1) orientado hacia el norte.
derecha	Instrucción primitiva que modifica la orientación del robot en 90° en sentido horario respecto de la orientación actual.
mover	Instrucción primitiva que conduce al robot de la esquina en la que se encuentra a la siguiente, respetando la dirección en la que está orientado. Es responsabilidad del programador que esta instrucción sea ejecutada dentro de los límites de la ciudad. En caso contrario se producirá un error y el programa será abortado.
tomarFlor	Instrucción primitiva que le permite al robot recoger una flor de la esquina en la que se encuentra y ponerla en su bolsa. Es responsabilidad del programador que esta instrucción sea ejecutada solo cuando haya al menos una flor en dicha esquina. En caso contrario se producirá un error y el programa será abortado.
tomarPapel	Instrucción primitiva que le permite al robot recoger un papel de la esquina en la que se encuentra y ponerlo en su bolsa. Es responsabilidad del programador que esta instrucción sea ejecutada solo cuando haya al menos un papel en dicha esquina. En caso contrario se producirá un error y el programa será abortado.
depositarFlor	Instrucción primitiva que le permite al robot depositar una flor de su bolsa en la esquina en la que se encuentra. Es responsabilidad del programador que esta instrucción sea ejecutada solo cuando haya al menos una flor en dicha bolsa. En caso contrario se producirá un error y el programa será abortado.

depositarPapel	Instrucción primitiva que le permite al robot depositar un papel de su bolsa en la esquina en la que se encuentra. Es responsabilidad del programador que esta instrucción sea ejecutada solo cuando haya al menos un papel en dicha bolsa. En caso contrario se producirá un error y el programa será abortado.
PosAv	Identificador que representa el número de avenida en la que el robot está actualmente posicionado. Su valor es un número entero en el rango 1..100 y no puede ser modificado por el programador.
PosCa	Identificador que representa el número de calle en la que el robot está actualmente posicionado. Su valor es un número entero en el rango 1..100 y no puede ser modificado por el programador.
HayFlorEnLaEsquina	Proposición atómica cuyo valor es V si hay al menos una flor en la esquina en la que el robot está actualmente posicionado, ó F en caso contrario. Su valor no puede ser modificado por el programador.
HayPapelEnLaEsquina	Proposición atómica cuyo valor es V si hay al menos un papel en la esquina en la que el robot está actualmente posicionado, ó F en caso contrario. Su valor no puede ser modificado por el programador.
HayFlorEnLaBolsa	Proposición atómica cuyo valor es V si hay al menos una flor en la bolsa del robot, ó F en caso contrario. Su valor no puede ser modificado por el programador.
HayPapelEnLaBolsa	Proposición atómica cuyo valor es V si hay al menos un papel en la bolsa del robot, ó F en caso contrario. Su valor no puede ser modificado por el programador.
Pos	Instrucción que requiere dos valores Av y Ca, cada uno de ellos en el rango 1..100, y posiciona al robot en la esquina determinada por el par (Av,Ca) sin modificar la orientación del robot.
Informar	Instrucción que permite visualizar en pantalla el contenido almacenado en alguna variable.

Tabla 4.1: Sintaxis del robot Visual DaVinci (III-LIDI, 2017).

Las instrucciones detalladas previamente permiten realizar acciones simples como contar, limpiar, depositar, entre otras. En algunos casos la acción realizada no refleja con exactitud la realidad, por ejemplo, al ejecutar la instrucción mover el robot avanza 1 cuadra trasladándose hacia la próxima esquina. Es de esperar que el recorrido de una cuadra en una ciudad esté representado por una serie de pasos. En otros casos se desconocen ciertos detalles de cómo es resuelta una acción, por ejemplo, se sabe que el robot tiene la capacidad de reconocer objetos como flores y papeles, pero se desconoce cómo se realiza este proceso de reconocimiento. Al mismo tiempo la modelización de la ciudad y los objetos distribuidos en ella tampoco se condice con una representación exacta de la realidad. Sin embargo, la modelización e instrucciones que se proveen son suficientes para cumplir los objetivos de enseñanza en un

curso de programación inicial permitiendo escribir programas y aprovechando las ventajas de un entorno gráfico e interactivo (De Giusti *et al.,* 2001).

4.2.2 El lenguaje del Robot

Los programas escritos en el lenguaje del robot tienen una estructura predeterminada con un conjunto de palabras clave que encuadran las distintas secciones del algoritmo:

1. Se dispone de un sector para los módulos.
2. Antes de comenzar con el programa se indican las variables del programa principal.
3. Finalmente, entre las palabras clave comenzar y fin se escribe el código del programa.

A continuación, se indica el esqueleto de un programa:

```
programa nombre_del_programa
procesos
        Sección de procesos.
        proceso nombre_del_proceso
        comenzar
                ....
                Código del proceso.
        fin
variables
        Sección de variables del programa principal.
comenzar
        Programa principal.
fin
```

El lenguaje posee un conjunto de reglas sintácticas bastante estrictas, principalmente en cuestiones de indentación y sensibilidad en el uso de mayúsculas y minúsculas. Las palabras clave, instrucciones y estructuras de control son case sensitive con lo cual deben respetar el uso de mayúsculas y minúsculas. Cuando su nombre está formado por más de una palabra, desde la segunda palabra en adelante cada una empieza con una letra mayúscula. A modo de ejemplo se puede observar la instrucción "mover" con una única palabra y la instrucción "tomarFlor" con más de una palabra en su nombre. Desde el punto de vista del desarrollo de algoritmos estas reglas pueden dificultar la tarea del alumno, sin embargo, el objetivo final es formar al estudiante en un buen estilo de programación. Se intenta mediante la aplicación de reglas sintácticas obtener un código altamente legible para facilitar la comprensión, adaptación y reutilización, entre otros (III-LIDI, 2017).

Se dispone de conectivos lógicos que permiten combinar proposiciones y poder expresar lógica de mayor complejidad. Se proveen los conectivos de negación (~), conjunción (&) y disyunción (|) con sus correspondientes tablas de verdad. Al momento de evaluar las condiciones el

operador ~ tiene mayor prioridad, seguido por & y | en igual prioridad. En el caso de tener varias conjunciones y/o disyunciones, la evaluación se realiza de izquierda a derecha. Es posible utilizar paréntesis para alterar el orden de evaluación.

El lenguaje contempla la utilización de variables en el programa principal y dentro de cada proceso. La sintaxis para declarar una variable es:

nombreVariable: tipoVariable

Existen 2 tipos de datos diferentes:

- Numero

  Representa números enteros sobre los que se puede aplicar operaciones aritméticas (+, -, *, /) y operaciones de relación de orden (=, <, >, <=, >=, <>).
- Boolean

  Representa un tipo de dato lógico que puede tomar el valor V (Verdadero) o F (Falso).

La asignación de un valor a una variable se realiza mediante el operador := y se debe tener en cuenta que las variables no son inicializadas automáticamente con un valor por defecto. En caso de consultar el valor de una variable que nunca fue inicializada se produce un error en ejecución (III-LIDI, 2017).

4.2.3 Estructuras de Control

El lenguaje provee las estructuras de control necesarias para introducir al alumno en la programación. Se dispone de las siguientes estructuras: (De Giusti *et al.,* 2001; III-LIDI, 2017)

1. Secuencia

   Es un conjunto de instrucciones que se ejecutaran una a continuación de otra.
2. Selección

   Permite seleccionar entre dos alternativas. Su sintaxis es:

   si (condición)
   instrucción o bloque de instrucciones a realizar si la condición es verdadera
   sino
   instrucción o bloque de instrucciones a realizar si la condición es falsa

   El sino puede omitirse en caso de ser necesario utilizando la siguiente sintaxis:

   si (condición)
   instrucción o bloque de instrucciones a realizar si la condición es verdadera
3. Repetición

   Permite ejecutar una acción o conjunto de acciones un número fijo de veces. Este número debe ser conocido de antemano. Su sintaxis es:

   repetir N

instrucción o bloque de instrucciones a realizar

4. Iteración

   Permite ejecutar una acción o conjunto de acciones hasta la ocurrencia de un valor de verdad determinado por una condición. Su sintaxis es:

   mientras (condición)

   instrucción o bloque de instrucciones a realizar mientras la condición es verdadera

Como puede observarse, se dispone de las estructuras de control básicas existentes en la mayoría de los lenguajes, permitiendo luego aplicar los conceptos aprendidos en aplicaciones reales.

4.2.4 Modularización

El lenguaje del robot soporta la utilización de procesos facilitando la resolución de problemas y aprovechando los beneficios de esta metodología (independencia entre módulos, facilitar la adaptación de código, reusabilidad y mantenimiento, entre otros). Un proceso tiene su propia sección de variables locales que no son conocidas por el programa principal, ni por el resto de los procesos. Al mismo tiempo, los procesos no conocen las variables del programa principal. La comunicación entre programa principal/proceso y proceso/proceso se realiza mediante la utilización de parámetros. El lenguaje admite la utilización de parámetros de entrada y de entrada/salida. El parámetro de entrada (también llamado por valor) es utilizado para recibir información. El parámetro de entrada/salida (también llamado por referencia) es utilizado para comunicar un dato en ambos sentidos, ya que un proceso recibe el dato, puede utilizarlo, modificarlo y luego retornar el nuevo valor al proceso emisor (De Giusti *et al.*, 2001; III-LIDI, 2017).

La sintaxis para la definición de un proceso es:

proceso *nombre_del_proceso (tipo_parametro nombre_parametro1:tipo_dato; tipo_parametro nombre_parametroN:tipo_dato)*
variables
    Sección de variables del proceso.
comenzar
    ….
    Código del proceso.
fin

4.2.5 El ambiente de Visual DaVinci

Hasta aquí se ha presentado al Robot junto con sus capacidades, la ciudad y objetos con los que interactúa. Se introdujo el lenguaje utilizado por el entorno, describiendo brevemente los conceptos informáticos que permite enseñar al alumno como son la programación estructurada, las estructuras de control, variables, conectivos lógicos, modularización y manejo de parámetros, entre otros. El ambiente de programación del robot es la herramienta (software) donde el alumno realiza las diferentes etapas en el desarrollo de algoritmos como son la codificación del algoritmo, su compilación, ejecución (a través de la visualización gráfica) y obtención de resultados para validar que el programa sea correcto. La herramienta Visual DaVinci posee las siguientes secciones: (Champredonde *et al.*, 1999; III-LIDI, 2017)

- Panel Principal

  El panel principal (ver Figura 4.2) está formado por un menú con las acciones necesarias para utilizar la aplicación. Entre estas acciones se encuentran operaciones básicas como crear, abrir, guardar e imprimir, entre otras. En la parte central se cuenta con aquellos íconos relacionados con la compilación, ejecución y detención del algoritmo. En el costado derecho se dispone de iconos para visualizar las distintas secciones que se detallan a continuación.

Figura 4.2: Visual DaVinci: panel principal.

- Ciudad

  Esta sección (ver Figura 4.3) permite observar el comportamiento del programa en ejecución. A medida que se ejecutan las instrucciones del algoritmo se visualiza gráficamente cómo avanza la ejecución hasta su finalización.

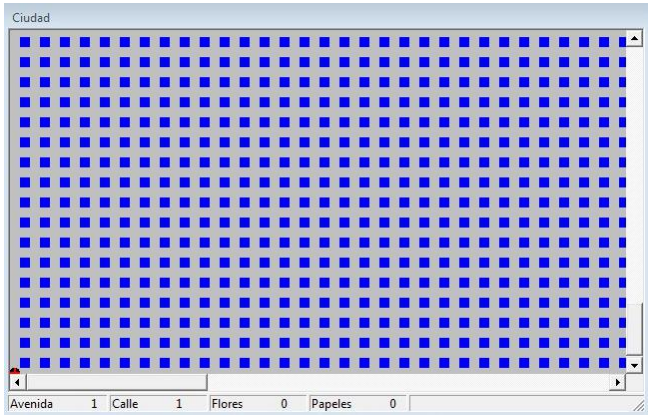


Figura 4.3: Visual DaVinci: ciudad.

- Editor de Código

  Esta sección (ver Figura 4.4) es un editor de texto con funcionalidad básica donde se escriben los algoritmos que luego son compilados y ejecutados.

Figura 4.4: Visual DaVinci: editor de código.

- Editor de Diagramas

  La sección Editor de Diagramas (ver Figura 4.5) permite implementar un algoritmo mediante la inserción de elementos que son seleccionados desde una barra de botones del editor.

Figura 4.5: Visual DaVinci: editor de diagramas.

- Coordenadas

  Posee la información directamente relacionada al robot como su posición (avenida/calle), dirección y cantidad de flores y papeles que cuenta en las bolsas. Esta información varía durante la ejecución del algoritmo reflejando el progreso (ver Figura 4.6).

Figura 4.6: Visual DaVinci: coordenadas.

En la Figura 4.7 se puede visualizar la aplicación Visual DaVinci:

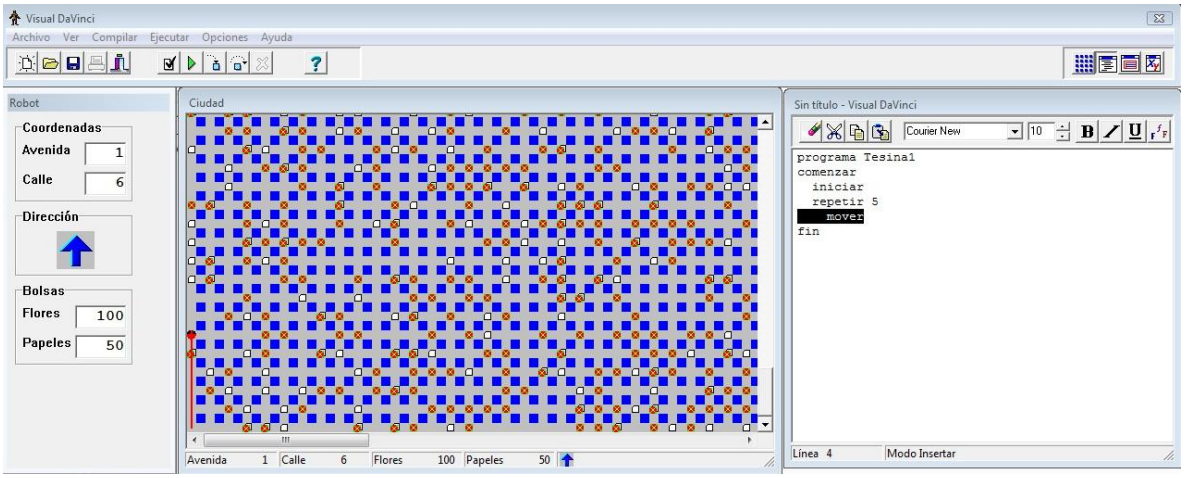


Figura 4.7: Visual DaVinci.

Se puede concluir que el entorno Visual DaVinci es una herramienta visual e interactiva que brinda un completo conjunto de funcionalidades para introducir al alumno en la programación de manera exitosa. La existencia de un lenguaje propio y acotado facilita la tarea del alumno sin limitar los conceptos que pueden enseñarse, ya que como se mencionó previamente Visual DaVinci contempla las estructuras de control básicas existentes en la mayoría de los lenguajes, el uso de variables, modularización y pasaje de parámetros, entre otros. Al ser una herramienta visual se logran obtener todos los beneficios planteados en el capítulo 3, y la simulación gráfica del algoritmo implementado en tiempo real a medida que es ejecutado, provee al alumno de un mecanismo de autocorrección. Al mismo tiempo, se reduce notablemente el grado de abstracción que necesita el alumno para comprender estos contenidos, ya que la visualización genera un contexto de aprendizaje más concreto.

4.3 CMRE

La herramienta Visual DaVinci presentada anteriormente, es utiliza desde hace muchos años en diversas cátedras de carreras de Informática de universidades nacionales, para abordar la

enseñanza y el aprendizaje de conceptos básicos de programación en los primeros años de la carrera. Los resultados obtenidos han sido exitosos y en consecuencia el entorno ha evolucionado hasta su versión actual conocida como CMRE (Concurrent Multi Robot Environment). CMRE es una herramienta desarrollada por el III-LIDI, cuya implementación ha sido realizada utilizando el lenguaje de programación Java (a diferencia de Visual DaVinci que fue implementado en Delphi). Esta evolución de Visual DaVinci mantiene la funcionalidad inicial (con algunos cambios que son tratados a continuación) e incorpora conceptos básicos de concurrencia y paralelismo. Como se mencionó en el capítulo 3, el avance tecnológico ha originado nuevos conceptos informáticos y es una tarea de las carreras informáticas incorporar estos temas en las currículas. CMRE es un entorno visual e interactivo que tiene como objetivo principal la enseñanza de conceptos de básicos de concurrencia en un curso inicial de algoritmos. Ha incorporado la utilización de múltiples robots que son interpretados como cores o procesadores, y al trabajar en conjunto simulan una arquitectura multiprocesador.

4.3.1 Características introducidas

El ambiente CMRE incorpora las siguientes características no contempladas en el Visual DaVinci: (De Giusti *et al.*, 2013; De Giusti *et al.*, 2015b; De Giusti *et al.*, 2016)

- Se cuenta con múltiples robots que realizan tareas dentro de la ciudad y pueden cooperar y/o competir entre ellos. Cada robot es visto como un procesador y el conjunto de robots es conceptualmente una arquitectura multiprocesador.
- La ciudad conformada por 100 avenidas y 100 calles puede ser subdividida en aéreas privadas, parcialmente compartidas y totalmente compartidas. En un área privada sólo puede trabajar un único robot. En un área parcialmente compartida se especifica un conjunto de robots que pueden trabajar en ella. Por último, en un área totalmente compartida, todos los robots declarados tendrán acceso y podrán trabajar en la misma.
- Es necesario poder sincronizar el trabajo de aquellos robots que trabajan simultáneamente sobre un área parcial o totalmente compartida. Por ejemplo, cuando dos o más robots intentan moverse hacia la misma esquina o acceder a los recursos (flores/papeles) que existan en cada esquina. Esta sincronización es resuelta mediante un mecanismo equivalente a un semáforo binario.
- Los robots pueden intercambiar información (datos o control) utilizando mensajes explícitos.
- Cada robot tiene una sección con la información de su posición (avenida y calle), la cantidad de flores y papeles existentes en la bolsa y su estado actual (ejecutando,

esperando la llegada de un mensaje o esperando por la liberación de una esquina, entre otros).

- Es posible seleccionar la velocidad con la trabajaran los robots (mínima, media o máxima).

Los ítems listados previamente son nuevas funcionalidades que se han agregado al entorno y tienen una correspondencia con un concepto de concurrencia y paralelismo que se desea enseñar al alumno (ver Tabla 4.2).

Concepto de Concurrencia y Paralelismo	**Funcionalidad CMRE**
Existencia de múltiples procesadores / cores.	Múltiples robots.
Memoria compartida.	Subdivisión de la ciudad en áreas compartidas a todos los robots.
Memoria distribuida.	Subdivisión de la ciudad en áreas exclusivas a un único robot.
Memoria compartida y distribuida.	Subdivisión de la ciudad en áreas parcialmente compartidas a un conjunto de robots.
Comunicación entre procesos por mensajes.	Intercambio de mensajes entre los robots.
Exclusión mutua sobre recursos compartidos.	Bloqueo de esquinas en la ciudad.
Exclusión mutua selectiva.	Acceso a áreas parcialmente compartidas de la ciudad.
Datos locales o globales.	Objetos numerables de la ciudad como las flores y los papeles.

Tabla 4.2: Correspondencia entre concepto de enseñanza y funcionalidad del entorno (De Giusti *et al.*, 2013).

4.3.2 El lenguaje CMRE

En base a las modificaciones que ha sufrido el entorno fue necesario realizar ajustes en el lenguaje utilizado por Visual DaVinci. La estructura predeterminada utilizada para escribir los programas en Visual DaVinci ha sido modificada por la siguiente: (De Giusti *et al.*, 2012a)

```
programa nombre_del_programa
procesos
        // Sección de procesos.
        proceso nombre_del_proceso
        variables
                // Variables del proceso
        comenzar
                // Código del proceso
        fin
areas
        nombreArea1: tipoArea(Coordenada0, Coordenada1, Coordenada2, Coordenada3)
        …
        nombreAreaN: tipoArea(Coordenada0, Coordenada1, Coordenada2, Coordenada3)
robots
        robot tipo1
        variables
                // Variables del robtot tipo1
        comenzar
                // Código del robot tipo1
        fin
        …
        robot tipoN
        variables
                // Variables del robtot tipoN
        comenzar
                // Código del robot tipoN
        fin
variables
        // Sección para declarar los robots
        nombreVariableRobot1: tipo1
        …
        nombreVariableRobotN: tipoN
comenzar
        // Se asignan las áreas de cada robot
        AsignarArea(nombreVariableRobot1, nombreArea1)
        …
        AsignarArea(nombreVariableRobotN, nombreAreaN)
        // Se inician los robots en la ciudad
        iniciar(nombreVariableRobot1, PosAv, PosCa)
        …
        iniciar(nombreVariableRobotN, PosAv, PosCa)
fin
```

Se puede observar que el entorno CMRE incorpora dos nuevas secciones a la estructura de un programa utilizando el lenguaje del robot. La sección "areas" donde se definen las subdivisiones de la ciudad, y la sección "robots" donde son declarados los robots que serán utilizados y que trabajarán sobre las distintas áreas definidas. Es necesario aclarar que el entorno CMRE puede ser utilizado en cursos iniciales de programación sin la necesidad de utilizar los conceptos de concurrencia y paralelismo incorporados en su versión actual. Para ello se debe declarar una única área que abarque toda la ciudad y un sólo robot.

Las instrucciones detalladas en la Tabla 4.1 se mantienen en el entorno CMRE, pero se incorpora un conjunto de nuevas instrucciones (ver Tabla 4.3).

Sintaxis	**Semántica**
AsignarArea(robot, area)	La instrucción asigna al robot un área de trabajo. Recibe dos parámetros: el robot y el área que debe asignarse.
iniciar(robot1, 1, 1)	Esta instrucción reemplaza al iniciar utilizado en Visual DaVinci. Notar que la existencia de múltiples robots en diferentes áreas requiere alterar su definición. En CMRE se deben especificar tres parámetros: el robot que será inicializado, el número de avenida y el número de calle donde será posicionado inicialmente el robot.
Random(valor, inf, sup)	Instrucción que calcula un número aleatorio entre los límites inferior y superior y retorna el resultado en el parámetro valor.
bloquearEsquina(avenida, calle)	Indica que el robot pide exclusión para la ocupación de una esquina. La instrucción recibe dos parámetros para identificar la esquina a bloquear: el número de avenida y el número de calle. Una vez que obtenga acceso a la esquina el robot podrá recoger o depositar objetos (flores/papeles).
liberarEsquina(avenida, calle)	Indica que el robot libera la esquina ocupada. La instrucción recibe dos parámetros para identificar la esquina a liberar: el número de avenida y el número de calle.
enviarMensaje(valor, robot)	Permite que un robot envíe un mensaje a otro. La instrucción recibe dos parámetros: el valor a recibir y el robot destinatario del mensaje. Al trabajar con un modelo asincrónico, el robot continúa con la siguiente instrucción secuencialmente sin esperar la recepción.
recibirMensaje(valor, robot)	Permite que un robot reciba un mensaje enviado por otro robot. La instrucción recibe dos parámetros: una variable para recibir el valor esperado y el robot desde el que se espera el mensaje. El robot que ejecuta esta instrucción queda a la espera hasta sincronizar con el robot que debe enviar el mensaje.

Tabla 4.3: Sintaxis del robot CMRE (De Giusti *et al.*, 2012a; De Giusti *et al.*, 2012b; III-LIDI, 2017).

El entorno CMRE incorpora las instrucciones de la tabla 4.3 para manejar dos cuestiones básicas en la ejecución concurrente: (De Giusti *et al.*, 2012a; De Giusti *et al.*, 2012b; De Giusti *et al.*, 2013)

- Manejo de colisiones

  La utilización de áreas compartidas por dos o más robots puede generar colisiones. En una esquina particular sólo puede trabajar un robot a la vez, esto significa que si una esquina se encuentra ocupada por el robot1 y el robot2 accede se produce una colisión. Para evitar este tipo de situaciones no deseadas, se provee la instrucción bloquearEsquina donde el robot solicita exclusión para ocupar una esquina (y poder recoger/depositar objetos) y la instrucción liberarEsquina para que el robot pueda liberar la esquina ocupada.
- Comunicación / Sincronización

  La incorporación de múltiples robots en la ciudad que trabajan de forma simultánea permite resolver un problema de manera colaborativa. El alumno puede resolver un determinado problema implementando un algoritmo que utilice varios robots que colaboran con un objetivo en común. Probablemente sea necesario que los robots se comuniquen y sincronicen para resolver el problema, por esta razón se han incorporado dos nuevas instrucciones. La instrucción enviarMensaje permite que un robot envíe un mensaje a otro y la instrucción recibirMensaje indica que un robot debe esperar hasta sincronizar con el envío de un mensaje proveniente desde otro robot.

4.3.3 El ambiente de CMRE

El ambiente de programación del robot ha evolucionado en base a las nuevas funcionalidades y conceptos de enseñanza incorporados a la herramienta. Se incorporaron nuevas secciones a las mencionadas en Visual DaVinci y otras fueron modificadas. El editor de diagramas fue eliminado por diferentes razones; principalmente porque su utilización para soportar todas las acciones que realiza el robot es cada vez más compleja. Desde el punto de vista educativo el editor de diagramas no brinda conceptos imprescindibles para la enseñanza. Por el contrario, se desea que el alumno aprenda a escribir sus propios algoritmos y no que el código sea autogenerado mediante la construcción de un diagrama gráfico. El entorno CMRE en su versión actual posee las siguientes secciones: (III-LIDI, 2017)

- Panel Principal

  En la Figura 4.8, se visualiza el panel o menú principal. Las acciones básicas para crear, guardar, abrir un programa y aquellas relacionadas con la compilación y ejecución del algoritmo (incluyendo pausar, ejecutar paso a paso, entre otras) se mantienen desde

Visual DaVinci. Se incorpora un ícono para indicar cuál será la velocidad (máxima, media o mínima) con la que ejecutaran los robots declarados, y otro con la imagen de una escoba utilizado para reiniciar el entorno (limpiando la ciudad y la bolsa del robot de flores/papeles).

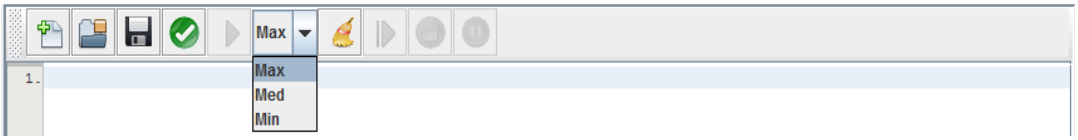


Figura 4.8: CMRE: panel principal.

- Ciudad

  La sección se mantiene desde Visual DaVinci con la diferencia que en CMRE se encuentra por debajo del editor de código (ver Figura 4.9).

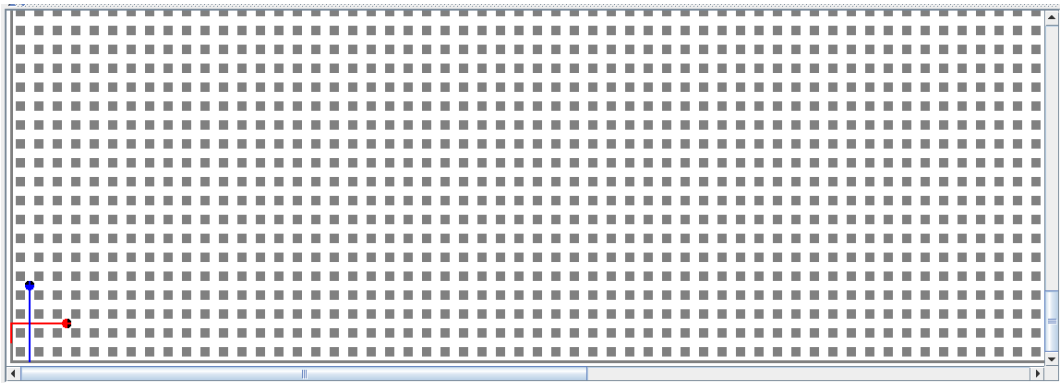


Figura 4.9: CMRE: ciudad.

- Miniatura

  Nueva sección que permite visualizar una versión en miniatura de la ciudad (ver Figura 4.10). En su interior posee un pequeño rectángulo que se corresponde con la parte de la ciudad que se está visualizando en la sección Ciudad durante la ejecución del algoritmo. Se permite desplazar el rectángulo dentro del cuadrado miniatura para observar diferentes aéreas de la sección ciudad, facilitando así la visualización de los robots y su comportamiento durante la ejecución.

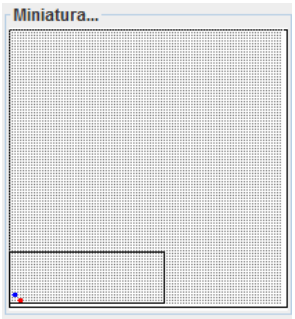


Figura 4.10: CMRE: miniatura.

- Editor de Código

  La sección permite al alumno escribir sus algoritmos para luego ejecutarlos (ver Figura 4.11). Se observa que la barra de herramientas disponible en Visual DaVinci para manejar cuestiones de formato (tamaño de letra, fuente, negrita, cursiva, entre otras) ha sido eliminada.

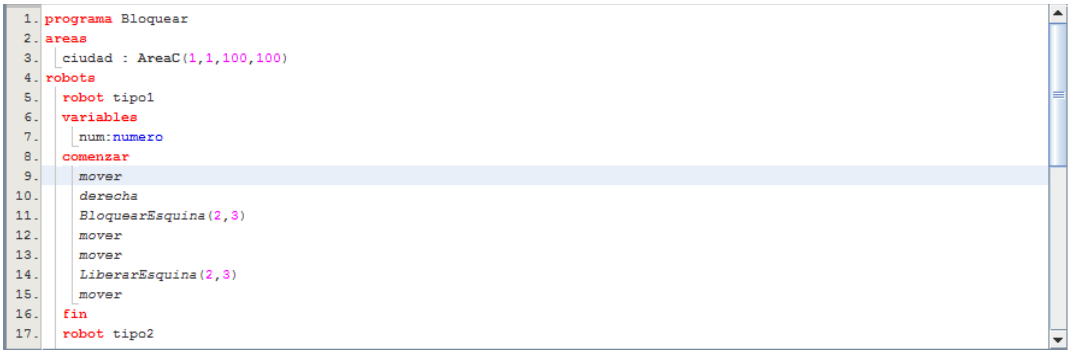


Figura 4.11: CMRE: editor de código.

- Detalles

  Nueva sección donde se permite al alumno configurar la distribución de flores y papeles en las esquinas de la ciudad y las cantidades iniciales en la bolsa de cada robot (ver Figura 4.12).

Figura 4.12: CMRE: detalles.

- Información del Robot

  Reemplaza la sección Coordenadas de Visual DaVinci. Notar que la capacidad de declarar múltiples robots requiere graficar la información de contexto por cada robot. En la Figura 4.13 se muestra un ejemplo con 2 robots donde se indica la dirección, posición (avenida y calle), cantidad de flores/papeles tanto en la bolsa como en la esquina donde se encuentra posicionado y una descripción que indica el estado (nuevo, ejecutándose, finalizado, entre otros).

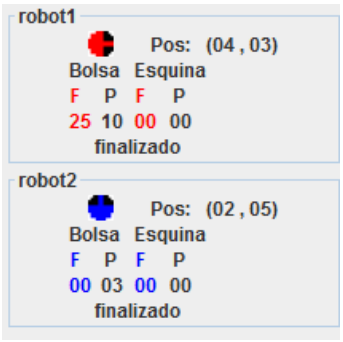


Figura 4.13: CMRE: información del robot.

En la Figura 4.14 se puede visualizar la aplicación CMRE:

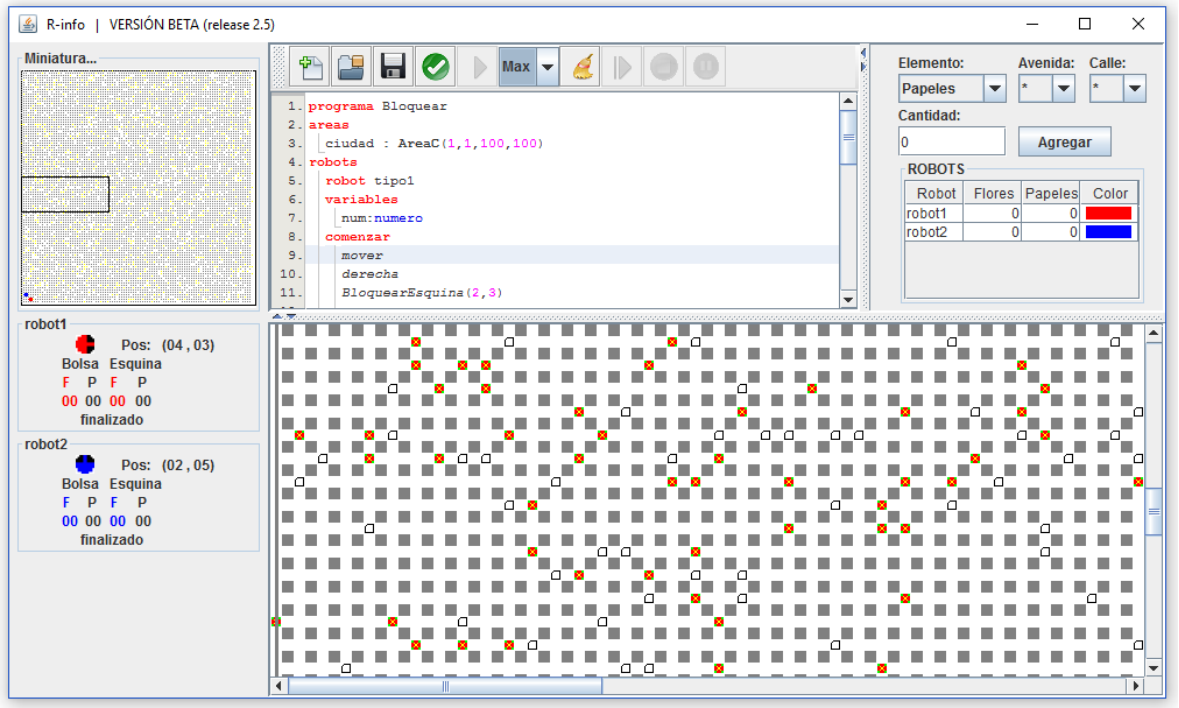


Figura 4.14: CMRE.

El entorno CMRE es una evolución de Visual DaVinci que incorpora conceptos básicos de concurrencia y paralelismo. Al igual que su predecesor aprovecha los beneficios de una herramienta visual e interactiva. En caso de ser necesario es posible emplear CMRE en un curso inicial de programación sin utilizar los conceptos de concurrencia que pueden resultar más complejos en una etapa inicial. Un caso concreto es la utilización de CMRE en el ingreso de la Facultad de Informática de la UNLP, donde los alumnos utilizan el entorno implementando algoritmos con un único robot y un conjunto de instrucciones reducido que no contempla las detalladas en la tabla 4.3. Desde el punto de vista de la usabilidad, la aplicación ha sufrido modificaciones; se han incorporado nuevas secciones que mejoran la interacción con la herramienta, como es el caso de las secciones Miniatura y Detalle. Se observa que, algunas funcionalidades existentes en Visual DaVinci y que no están relacionadas con la enseñanza de

conceptos de programación (como es el caso de la barra de herramientas en el editor de código) no han sido migradas a CMRE.

4.4 Resumen del Capítulo

En el capítulo 4 se ha presentado el entorno CMRE (Concurrent Multi Robot Environment) en su versión actual, el cuál incorpora conceptos básicos e concurrencia y paralelismo para la enseñanza. Previamente, fue necesario describir el entorno Visual DaVinci ya que es el punto de partida para la migración a CMRE y donde nacen las bases de la aplicación, como es la definición de una ciudad conformada por avenidas y calles donde un robot se traslada e interactúa con objetos como son las flores/papeles. Visual DaVinci posee un lenguaje de programación propio con un conjunto de instrucciones acotado para facilitar el aprendizaje de los alumnos, sin embargo, el grupo de conceptos que permite enseñar es muy completo y cumple con creces las necesidades de un curso de programación inicial. En este capítulo, se describen aquellas funcionalidades introducidas en CMRE (como la utilización de múltiples robots, las velocidades, la división de la ciudad en aéreas de trabajo, entre otras), los cambios en el lenguaje del robot y la incorporación de nuevas secciones al ambiente. Como se detalló en la tabla 4.2, estas modificaciones tienen una correspondencia con un concepto teórico como las arquitecturas multiprocesador, memoria compartida/distribuida, exclusión mutua, entre otros.

En el capítulo 5, se define e implementa un prototipo que es una extensión del entorno CMRE llamada ECMRE (Extended Concurrent Multi Robot Environment) y que incorpora a la herramienta, aquellos conceptos del capítulo 2 entre los que se encuentran los procesadores multicore y sus arquitecturas, el consumo de energía, la temperatura de los procesadores y el balance de carga de la aplicación, entre otros. La extensión ECMRE integra nuevos conceptos que han tomado mayor relevancia en los últimos años con el objetivo de utilizar la herramienta en las asignaturas de los primeros años de la Facultad de Informática de la UNLP.

CAPITULO 5
ECMRE (EXTENDED CONCURRENT MULTI ROBOT ENVIRONMENT)

5.1 Introducción

En el capítulo 4 se ha presentado el entorno CMRE (Concurrent Multi Robot Environment) en su versión actual, utilizado en las asignaturas de los primeros años de las carreras de la Facultad de Informática de la UNLP. Se describieron cuales son los conceptos de programación básicos que se abordan tales como la programación estructurada, estructuras de control, variables, conectivos lógicos, modularización y manejo de parámetros, entre otros. También fueron detallados aquellos conceptos de concurrencia y paralelismo que se presentan en el entorno mediante la utilización de múltiples robots y la subdivisión de secciones en la ciudad.

En este capítulo, se define e implementa un prototipo llamado ECMRE (Extended Concurrent Multi Robot Environment) que incorpora los conceptos de arquitecturas multicore (homogéneas y heterogéneas), balance de carga, el consumo energético y la temperatura de los procesadores, presentados en el capítulo 2. La aplicación CMRE ha sido desarrollada en lenguaje Java y utiliza las bibliotecas Swing y AWT para la resolución de la interfaz gráfica. La ampliación funcional que se propone en ECMRE incorpora a los recursos ya mencionados la utilización de la librería jfreechart que se utiliza para la creación de gráficos de barra, histogramas, entre otros. Además, en ECMRE se permite al alumno configurar diferentes tipos de arquitecturas multicore y observar en forma gráfica e interactiva cómo varían los valores de consumo y temperatura durante la ejecución de su algoritmo. Uno de los ejes principales del prototipo es la temperatura y el consumo energético de los procesadores; por esta razón se incluyen mecanismos para incrementar/disminuir la velocidad del procesador durante la ejecución de un algoritmo, de acuerdo al aumento de consumo energético, e incluso detener a algún robot/procesador cuando su temperatura alcanza un valor superior al límite máximo configurado. Estos mecanismos intentan reflejar el comportamiento real de los procesadores contemporáneos. Para la comprensión y el análisis de lo ocurrido durante la ejecución del algoritmo se incorporó, al entorno, una sección con tablas y gráficos que detallan el trabajo realizado por cada robot/procesador desde el punto de vista del consumo de energía y temperatura. Mediante el análisis de esta información, el alumno podrá modificar su algoritmo (balanceando la carga de acuerdo a la arquitectura multicore actual) para obtener soluciones que resulten eficientes desde el punto de vista de la energía y temperatura utilizadas.

5.2 Representación de arquitecturas multicore en ECMRE

Los procesadores actuales presentan una arquitectura paralela. Los procesadores multicores integran múltiples cores o núcleos en un único procesador físico. En el capítulo 2 se describieron las arquitecturas multicore y su clasificación en base las características de sus núcleos (homogénea o heterogénea) y como se expresa en el capítulo 4, el entorno CMRE utiliza múltiples robots que son interpretados como cores o procesadores, que al trabajar en conjunto simulan una arquitectura multiprocesador. Asimismo, en el entorno CMRE se trabaja con el concepto de velocidad del robot (mínima, media o máxima), sin embargo, en cada ejecución todos los robots operan a la misma velocidad permitiendo representar sólo arquitecturas multiprocesador homogéneas.

En una arquitectura heterogénea los cores poseen distintas características en cuanto a rendimiento y consumo de energía. La extensión realizada en ECMRE incorpora la posibilidad de declarar robots que trabajan a distinta velocidad durante la ejecución del algoritmo. Cada robot posee un tiempo, consumo energético y temperatura que son actualizados a medida que se ejecuta cada instrucción. Del conjunto de primitivas que posee el lenguaje del robot, se ha decidido trabajar con un subconjunto que incluye aquellas instrucciones de mayor relevancia en la ejecución. Son descartadas instrucciones como AsignarArea e Iniciar, ya que se ejecutan una única vez por cada procesador y no son determinantes en relación a la totalidad del trabajo realizado. El subconjunto de instrucciones a considerar incluye:

1. bloquearEsquina(avenida, calle)
2. depositarFlor
3. depositarPapel
4. derecha
5. enviarMensaje(valor, robot)
6. liberarEsquina(avenida, calle)
7. mover
8. recibirMensaje(valor, robot)
9. tomarFlor
10. tomarPapel

5.2.1 Rendimiento de un procesador

Las arquitecturas heterogéneas están conformadas por cores de distinto rendimiento. La limitación del entorno CMRE reflejada en relación a que todos los robots deben trabajar a igual velocidad, hace que el rendimiento sea el mismo en todos los cores/procesadores.

En ECMRE se ha modificado la sección Detalles (ver Figura 5.1) agregando a la tabla ROBOTS la característica relacionada con la velocidad. De esta forma se permite trabajar con robots que pueden variar en cuanto a rendimiento dado que ejecutarán instrucciones a distinta velocidad.

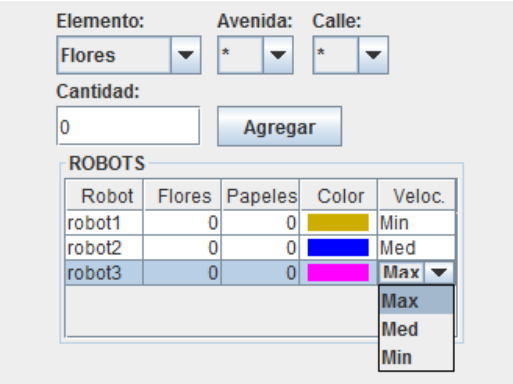


Figura 5.1: ECMRE - Información detallada de cada robot.

Como se muestra en la Figura 5.1 existen 3 velocidades y la relación entre ellas es la siguiente:

- Max es la máxima velocidad disponible. Un robot que trabaja en Max obtiene un mejor rendimiento que en velocidades Med o Min.
- Un robot que trabaja a velocidad media (Med) lo hace a la mitad de la velocidad establecida como velocidad máxima.
- Un robot que ejecuta en velocidad mínima (Min) trabaja a la mitad de la velocidad establecida como velocidad media y por lo tanto, a la cuarta parte de la velocidad máxima (Max).

Otro punto a mejorar, en CMRE, es la cuantificación del tiempo de ejecución para poder realizar un análisis de rendimiento. Al ejecutar sus algoritmos, el alumno puede observar que la velocidad incide sobre el tiempo final. Por ejemplo, un algoritmo con robots trabajando a máxima velocidad obtiene un menor tiempo de ejecución que si es ejecutado con robots a velocidad media. Sin embargo, no se dispone de valores concretos y se desconoce el tiempo consumido por cada instrucción.

Para mejorar este aspecto, en ECMRE se define T como unidad de tiempo (equivale a 100 milisegundos) para medir el rendimiento de los robots/procesadores. Como se mencionó anteriormente, se ha decidido trabajar con un subconjunto de 10 instrucciones primitivas. Para ello se realizó un relevamiento de cuáles son las acciones que involucra cada instrucción primitiva en el lenguaje del robot y en base a los resultados obtenidos se asignaron T unidades de tiempo a cada instrucción. Estas instrucciones primitivas fueron clasificadas en 3 categorías que reflejan su grado de complejidad (ver Tabla 5.1) y sobre ellas se asignó un tiempo de ejecución (ver Tabla 5.2), consecuente con la velocidad del robot:

Instrucción	Descripción	Complejidad
bloquearEsquina(avenida, calle)	Se evalúa que los parámetros ingresados representen una posición válida, se solicita el acceso y luego se bloquea la esquina. Mientras exista el bloqueo la esquina no puede ser utilizada por otro robot.	ALTA
depositarFlor	Se incrementa en 1 la cantidad de flores en la esquina donde se encuentra posicionado el robot. En caso que el robot no posea flor en la bolsa se informa un error. Se resta una flor en la bolsa del robot.	MEDIA
depositarPapel	Se incrementa en 1 la cantidad de papeles en la esquina donde se encuentra posicionado el robot. En el caso que el robot no posea papel en la bolsa se informa un error. Se resta un papel en la bolsa del robot.	MEDIA
derecha	Se modifica la orientación del robot.	BAJA
enviarMensaje(valor, robot)	El robot origen toma el valor ingresado por parámetro y se lo envía al robot destino.	ALTA
liberarEsquina(avenida, calle)	Se evalúa que los parámetros ingresados representen una posición válida y se libera la esquina.	ALTA
mover	Se modifica la posición del robot avanzando 1 cuadra en base a la orientación actual. Se verifica que la posición destino esté dentro del área válida.	BAJA
recibirMensaje(valor, robot)	El robot queda a la espera de un mensaje proveniente del robot recibido por parámetro.	ALTA
tomarFlor	Se resta una flor de la cantidad de flores de la esquina donde se encuentra posicionado el robot. En el caso que la esquina no posea flor se informa un error. Se incrementa en 1 la cantidad de flores en la bolsa del robot.	MEDIA
tomarPapel	Se resta un papel de la cantidad de papeles de la esquina donde se encuentra posicionado el robot. En el caso que la esquina no posea papel se informa un error. Se incrementa en 1 la cantidad de papeles en la bolsa del robot.	MEDIA

Tabla 5.1: Grado de complejidad asignado a instrucciones primitivas.

Notar en la Tabla 5.2 que el tiempo por complejidad se duplica al disminuir la velocidad del robot.

Complejidad	T_{Max}	T_{Med}	T_{Min}
BAJA	1	2	4
MEDIA	2	4	8
ALTA	3	6	12

Tabla 5.2: Asignación de unidades de tiempo por complejidad y velocidad.

Como se muestra en la Figura 5.2, se ha incorporado a la sección Información del robot, el tiempo de ejecución en T unidades. El mismo se actualiza durante la ejecución y es de gran utilidad para evaluar la performance del algoritmo.

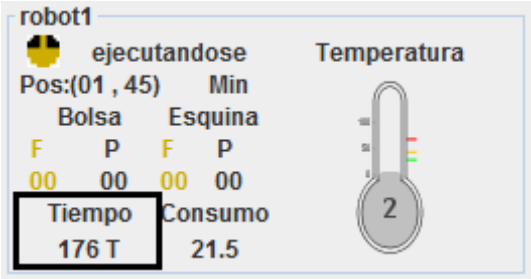


Figura 5.2: ECMRE: información del robot - tiempo de ejecución.

El manejo de velocidad por robot y la asignación de tiempos a las instrucciones primitivas en ECMRE introduce la posibilidad de trabajar con robots/procesadores que poseen distinto rendimiento, que es una de las características de una arquitectura multicore heterogénea.

5.2.1.1 Velocidad por robot: ejemplo práctico

A continuación, se plantea un caso experimental sencillo para mostrar el impacto que tiene la utilización de robots que trabajan a distinta velocidad, en relación a su rendimiento.

Problema 1: Se tienen 3 robots que ejecutan a diferentes velocidades (Max, Med y Min). Cada robot debe recorrer una avenida completa, por ejemplo, las avenidas 1, 2 y 3 (Av1, Av2 y Av3 respectivamente).

El tiempo de la instrucción mover es 1T (Max), 2T (Med) y 4T (Min), si se tiene en cuenta su complejidad. Según el trabajo esperado, se tiene:

- R1 en Max recorre la Av1 y tarda 99 * 1 T = 99 T
- R2 en Med recorre la Av2 y tarda 99 * 2 T = 198 T
- R3 en Min recorre la Av3 y tarda 99 * 4 T = 396 T

Como se muestra en la Figura 5.3, el tiempo de ejecución por robot ha sido 99 T para R1, 198 T para R2 y 396 T para R3. El algoritmo ha tenido un tiempo de ejecución total de 396 T que se corresponde con el tiempo de R3.

Figura 5.3: Resultado problema 1 - tiempos de ejecución por robot.

5.2.2 Consumo de energía de un procesador

Otra característica de una arquitectura heterogénea es la diferencia de consumo energético que tienen los cores involucrados. Como se explicó en el capítulo 2, el consumo energético de los procesadores es un tema que ha tomado importancia en los últimos años. La performance de los algoritmos no sólo es evaluada por el tiempo de ejecución, sino también por la energía consumida.

El entorno ECMRE incorpora una nueva sección llamada Robot / Procesador donde el alumno puede parametrizar cuál es el consumo energético (en Joules) para el conjunto de instrucciones primitivas detallado en la Tabla 5.1. Al momento de compilar el algoritmo, se asigna un valor de consumo por defecto para cada instrucción, que puede ser modificado por el alumno (antes de ejecutar su algoritmo) para representar diferentes escenarios.

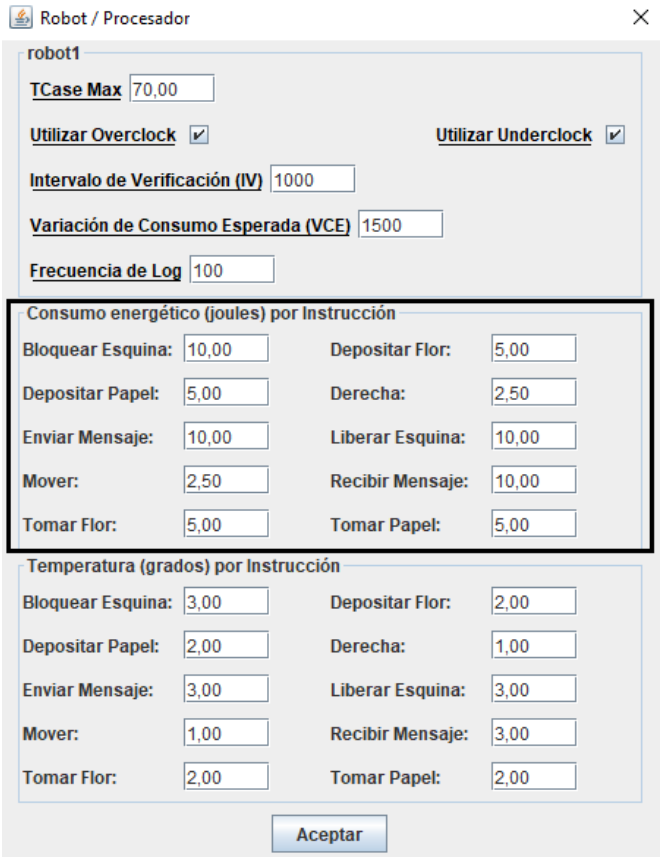


Figura 5.4: ECMRE - Consumo energético por instrucción en sección Robot / Procesador.

Se ha ampliado la lógica existente en CMRE para que cada robot almacene el consumo correspondiente y que el mismo pueda ser actualizado durante la ejecución del algoritmo. En la Figura 5.5 se observa la sección información del robot donde se aprecia el consumo energético del mismo.

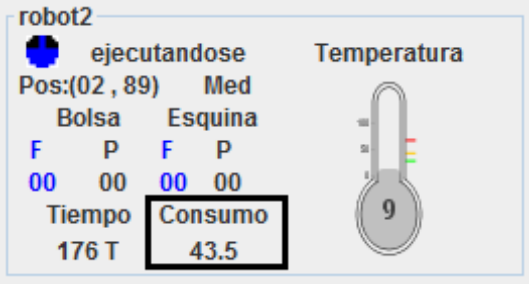


Figura 5.5: ECMRE - información del robot: consumo energético.

La velocidad del procesador es un factor que incide sobre el consumo (a mayor velocidad, mayor es el consumo). En ECMRE se define una constante por velocidad que es utilizada para actualizar el consumo del robot, y su valor es 0.25 para Min, 0.5 para Med y 1 para Max. Esta constante es fija y no puede ser modificada por el usuario del entorno. Su valor ha sido definido para representar con claridad los conceptos incorporados a ECMRE en base a la implementación de los algoritmos de cálculo que actualizan el estado del procesador y también

del análisis realizado a partir de numerosos casos de estudio. Por cada instrucción ejecutada el consumo se calcula como:

$$\boldsymbol{Consumo_{Robot} = Consumo_{Robot} + (consumo_{instrucción} * constC_{vel})} \quad \textit{(Fórmula 5.1)}$$

Notar que el consumo por instrucción ($consumo_{instrucción}$) es el valor parametrizado en la sección Robot / Procesador y que la constante de consumo por velocidad ($constC_{vel}$), del robot al momento de ejecutar la instrucción, es 1, 0.5 o 0.25 dependiendo de la velocidad Max, Med y Min respectivamente.

La capacidad de trabajar con robots/procesadores que poseen distinto consumo energético es otra de las características de una arquitectura multicore heterogénea, y que ha sido introducida en ECMRE.

5.2.2.1 Consumo por robot: ejemplo práctico

A continuación, se plantea un caso experimental para mostrar cómo se calcula el consumo de cada robot durante la ejecución del algoritmo.

Problema 2: Se tienen 3 robots que ejecutan a velocidad Max, Med y Min. Cada robot debe recorrer una avenida completa, por ejemplo, Av1, Av2 y Av3 y recoger 10 flores que seguro existen.

Dada la configuración de consumo por instrucción de cada robot, como se muestra en la Figura 5.6:

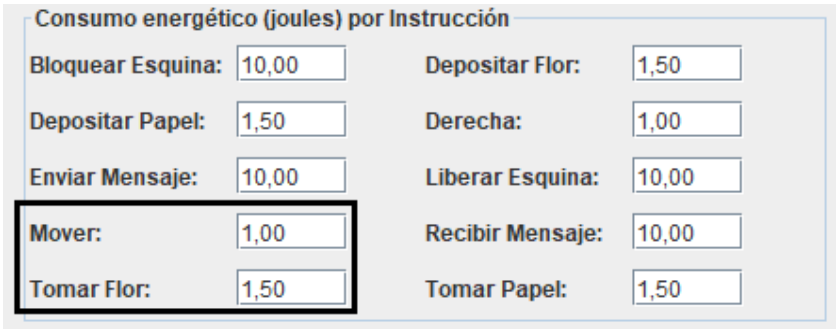


Figura 5.6: Problema 2 - configuración de consumo.

El consumo por robot para la ejecución del algoritmo es:

- R1 en Max consume un total de 114 Joules que se obtienen de:
  - 99 mover → 99 * 1 Joule * 1 → 99 Joules
  - 10 tomarFlor → 10 * 1.5 Joules * 1 → 15 Joules
- R2 en Med consume un total de 57 Joules que se obtienen de:

- 99 mover → 99 * 1 Joule * 0.5→ 49.5 Joules
- 10 tomarFlor → 10 * 1.5 Joules * 0.5 → 7.5 Joules

- R3 en Min consume un total de 28.5 Joules que se obtienen de:
  - 99 mover → 99 * 1 Joule * 0.25 → 24.75 Joules
  - 10 tomarFlor → 10 * 1.5 Joules * 0.25 → 3.75 Joules

Como se muestra en la Figura 5.7, el consumo total para la ejecución del algoritmo es 114 + 57 + 28.5 = 199.5 Joules. Notar que en este caso se ha utilizado la misma configuración de consumo por instrucción para todos los robots. Sin embargo, es posible indicar diferentes valores de consumo para la misma instrucción por cada procesador.

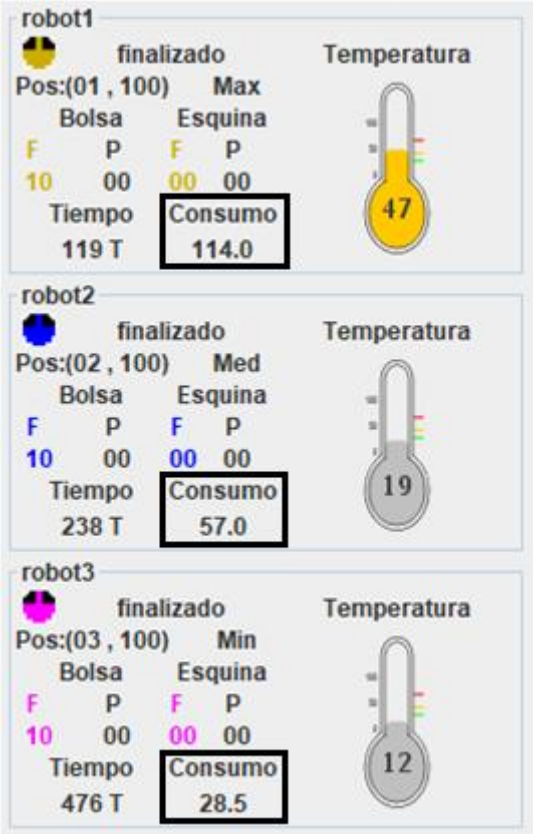


Figura 5.7: Resultado problema 2 - consumo energético por robot.

5.2.3 Temperatura de un procesador

La temperatura del procesador es otro de los temas tratados en el capítulo 2. Los procesadores de gran rendimiento generan altas temperaturas pudiendo caer en situaciones no deseadas que afectan tanto al hardware como software. Un procesador expuesto a temperaturas superiores a sus especificaciones máximas puede sufrir daños físicos, disminuir su tiempo de vida útil, tener fallas en su funcionamiento, como por ejemplo, la generación incorrecta de datos, e incluso el bloqueo de la totalidad del sistema. Con el objetivo de evitar estas deficiencias, los fabricantes

de hardware han incorporado técnicas que ajustan el rendimiento del procesador en base a su temperatura.

ECMRE incorpora el concepto de temperatura por robot/procesador. Para ello al igual que ocurre con el consumo, el alumno puede configurar en la sección Robot / Procesador (ver Figura 5.8) la temperatura del robot/procesador, expresada en grados, que corresponde a cada una de las instrucciones del conjunto de primitivas descriptas en la Tabla 5.1. Al momento de compilar el algoritmo, se asigna un valor de temperatura por defecto para cada instrucción, que puede ser modificado por el alumno (antes de ejecutar su algoritmo) simulando diferentes escenarios.

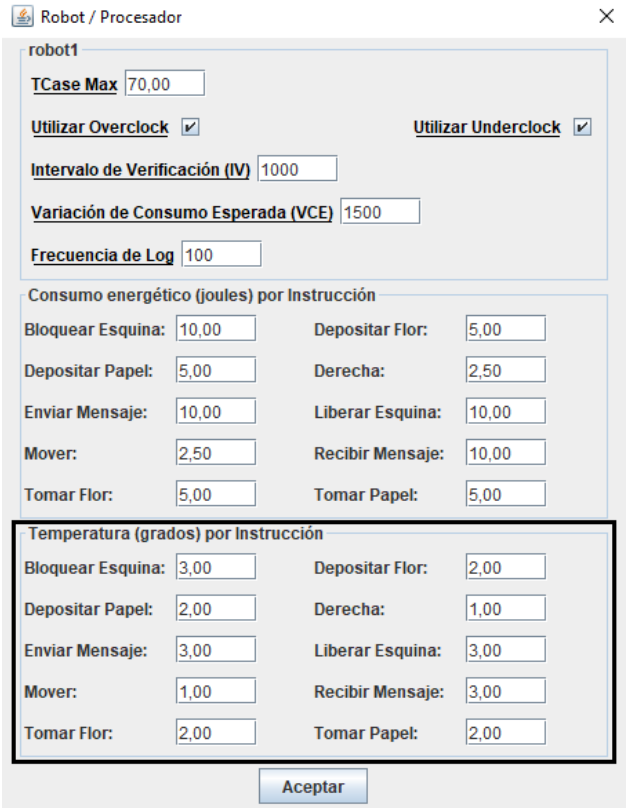


Figura 5.8: ECMRE - Temperatura por instrucción en sección Robot / Procesador.

En ECMRE se ha implementado la lógica necesaria para que cada robot registre su temperatura, que la misma sea actualizada y a la vez pueda visualizarse durante la ejecución del algoritmo. Para ello, se incorporó un termómetro en la sección Información del robot (ver Figura 5.9) que refleja la temperatura con un valor numérico. El termómetro cambia de color (gris, verde, naranja y rojo) a medida que disminuye o aumenta la temperatura, representando con el color gris a la temperatura mínima y con rojo a la máxima.

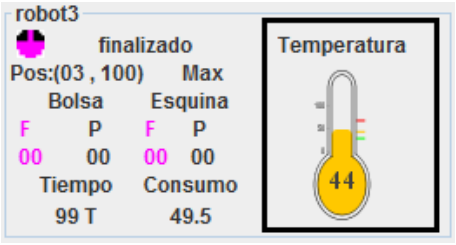


Figura 5.9: ECMRE - información del robot: temperatura.

La velocidad del procesador es un factor que incide sobre la temperatura que el mismo alcanza, lo que indica que, a mayor velocidad, mayor es el incremento de temperatura. Se define una constante por velocidad que es utilizada para actualizar la temperatura del robot, y su valor es 0.92 para velocidad Min, 0.95 para Med y 0.98 para Max. Su valor ha sido definido para representar con claridad los conceptos incorporados a ECMRE en base a la implementación de los algoritmos de cálculo que actualizan el estado del procesador y al análisis realizado a partir de los casos de estudio tratados. Por cada instrucción ejecutada, la temperatura se calcula como:

$$\boldsymbol{Temp_{Robot} = (temp_{instrucción} + Temp_{Robot}) * constT_{temp}} \qquad \textit{(Fórmula 5.2)}$$

Notar que la temperatura por instrucción ($temp_{instrucción}$) es el valor parametrizado en la sección Robot / Procesador y que la constante de temperatura por velocidad ($constT_{temp}$), dependiendo de la velocidad del robot al momento de ejecutar la instrucción, es 0.92, 0.95 y 0.98, respectivamente.

5.2.3.1 Temperatura por robot: ejemplo práctico

A continuación, se plantea un caso experimental sencillo para mostrar el incremento de la temperatura durante la ejecución de un algoritmo.

Problema 3: Se tienen 2 robots que ejecutan a velocidades Max y Min. Cada robot debe realizar un cuadrado de 2 cuadras x 2 cuadras depositando un papel en cada esquina. El robot 1 inicia su recorrido en la posición (1,1), el robot 2 en (5,1) y ambos tienen suficiente cantidad de papeles para realizar lo pedido.

Dada la configuración de grados por instrucción de cada robot, que se muestra en la Figura 5.10:

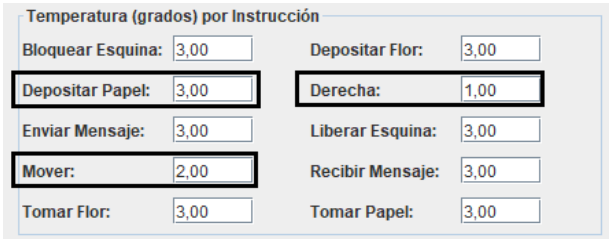


Figura 5.10: Problema 3 - configuración de temperatura.

R1 trabaja a velocidad Max. Inicia con una temperatura de 0° y durante la ejecución del programa se visualiza la siguiente variación:

1)	mover	→	(2 + 0.00) * 0.98	→ 1.96°
2)	depositarPapel	→	(3 + 1.96) * 0.98	→ 4.861°
3)	mover	→	(2 + 4.861) * 0.98	→ 6.724°
4)	depositarPapel	→	(3 + 6.724) * 0.98	→ 9.529°
5)	derecha	→	(1 + 9.529) * 0.98	→ 10.319°
6)	mover	→	(2 + 10.319) * 0.98	→ 12.072°
7)	depositarPapel	→	(3 + 12.072) * 0.98	→ 14.771°
8)	mover	→	(2 + 14.771) * 0.98	→ 16.435°
9)	depositarPapel	→	(3 + 16.435) * 0.98	→ 19.047°
10)	derecha	→	(1 + 19.047) * 0.98	→ 19.646°
11)	mover	→	(2 + 19.646) * 0.98	→ 21.213°
12)	depositarPapel	→	(3 + 21.213) * 0.98	→ 23.728°
13)	mover	→	(2 + 23.728) * 0.98	→ 25.214°
14)	depositarPapel	→	(3 + 25.214) * 0.98	→ 27.65°
15)	derecha	→	(1 + 27.65) * 0.98	→ 28.077°
16)	mover	→	(2 + 28.077) * 0.98	→ 29.475°
17)	depositarPapel	→	(3 + 29.475) * 0.98	→ 31.826°
18)	mover	→	(2 + 31.826) * 0.98	→ 33.149°
19)	depositarPapel	→	(3 + 33.149) * 0.98	→ 35.426°
20)	derecha	→	(1 + 35.426) * 0.98	→ 35.698°

R2 trabaja a velocidad Min. Inicia con una temperatura de 0° y durante la ejecución del programa se visualiza la siguiente variación:

1)	Mover	→	(2 + 0.00) * 0.92	→ 1.84°
2)	depositarPapel	→	(3 + 1.84) * 0.92	→ 4.453°
3)	Mover	→	(2 + 4.453) * 0.92	→ 5.937°
4)	depositarPapel	→	(3 + 5.937) * 0.92	→ 8.222°
5)	Derecha	→	(1 + 8.222) * 0.92	→ 8.484°
6)	Mover	→	(2 + 8.484) * 0.92	→ 9.645°
7)	depositarPapel	→	(3 + 9.645) * 0.92	→ 11.634°
8)	Mover	→	(2 + 11.634) * 0.92	→ 12.543°
9)	depositarPapel	→	(3 + 12.543) * 0.92	→ 14.299°
10)	Derecha	→	(1 + 14.299) * 0.92	→ 14.076°
11)	Mover	→	(2 + 14.076) * 0.92	→ 14.789°
12)	depositarPapel	→	(3 + 14.789) * 0.92	→ 16.366°
13)	Mover	→	(2 + 16.366) * 0.92	→ 16.897°
14)	depositarPapel	→	(3 + 16.897) * 0.92	→ 18.305°
15)	Derecha	→	(1 + 18.305) * 0.92	→ 17.761°
16)	mover	→	(2 + 17.761) * 0.92	→ 18.18°
17)	depositarPapel	→	(3 + 18.18) * 0.92	→ 19.486°
18)	mover	→	(2 + 19.486) * 0.92	→ 19.767°
19)	depositarPapel	→	(3 + 19.767) * 0.92	→ 20.945°
20)	derecha	→	(1 + 20.945) * 0.92	→ 20.19°

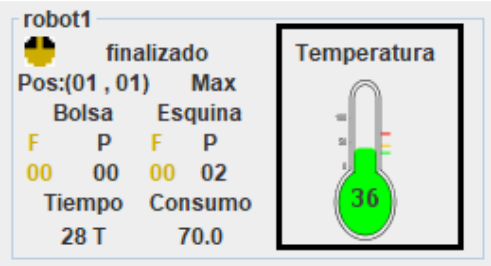


Figura 5.11: Resultado problema 3 - temperatura robot1.

Figura 5.12: Resultado problema 3 - temperatura robot2.

Como se observa en las Figuras 5.11 y 5.12, la temperatura alcanzada al finalizar la ejecución ha sido de 36 grados para el robot1 y 20 de grados para el robot2. En este ejemplo, la diferencia se origina a partir de la constante de temperatura por velocidad, cuyo valor es 0.92 para velocidad Min, 0.95 para Med y 0.98 para Max.

5.3 Técnicas de ajuste de rendimiento en ECMRE

La incorporación en ECMRE del tiempo de ejecución, consumo y temperatura presentados anteriormente permiten contar con información del estado del procesador (robot) durante la ejecución de un algoritmo. En el capítulo 2 se describieron algunas técnicas de ajuste de rendimiento que utilizan los procesadores actuales tomando el consumo energético, la temperatura, y otros valores como indicadores para toma de decisiones. Overclocking y underclocking son dos de las técnicas más utilizadas para aumentar o disminuir la frecuencia del clock del procesador con el objetivo de incrementar la performance o normalizar los valores de consumo y temperatura cuando el procesador se encuentra sobrecargado.

ECMRE contempla la posibilidad de que los robots apliquen overclocking y underclocking durante la ejecución del algoritmo. En la sección Robot/Procesador se configuran los siguientes parámetros que activan dicha funcionalidad (Figura 5.13):

1. TCase Max

   Es la temperatura máxima del procesador. Si se sobrepasa este valor:

   a. Se intentará aplicar underclock para disminuir la temperatura del procesador.
   b. Si no se admite underclock o la velocidad es la mínima, el robot se detiene hasta normalizar su temperatura.
2. Utilizar Overclock

   Marca que habilita la utilización de overclocking en el robot.
3. Utilizar Underclock

   Marca que habilita la utilización de underclocking en el robot.
4. Intervalo de Verificación (IV)

   Indica la cantidad de unidades T que posee el intervalo de tiempo utilizado para verificar el incremento del consumo energético y aplicar overclocking/underclocking según corresponda.
5. Variación de Consumo Esperada (VCE)

   Indica la variación de consumo energético (en joules) esperada para el tiempo definido en IV. Valor utilizado para aplicar overclocking/underclocking.

Figura 5.13: ECMRE - Parámetros de ajuste de rendimiento en sección Robot / Procesador.

El usuario puede hacer click sobre el nombre del parámetro para ver una descripción funcional del mismo. En la Figura 5.14 se visualiza el texto correspondiente al parámetro TCase Max.

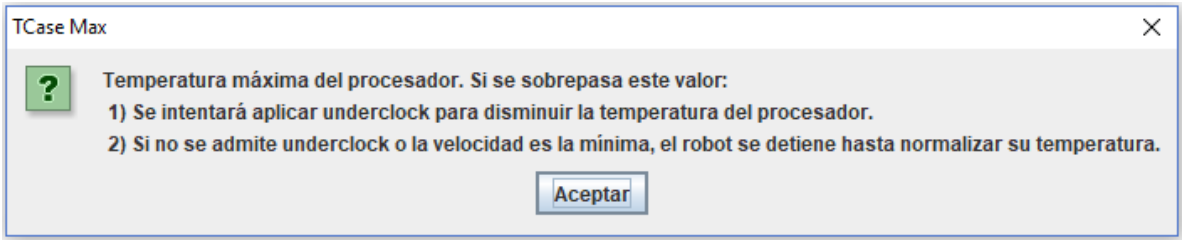


Figura 5.14: ECMRE - Descripción funcional del parámetro TCase Max.

5.3.1 Overclocking & Underclocking

Se ha implementado la lógica necesaria para evaluar el estado del robot/procesador durante la ejecución y aplicar overclocking/underclocking. El algoritmo de ajuste de rendimiento trabaja con los parámetros de configuración mencionados previamente (Figura 5.13) y una serie de atributos pertenecientes a cada procesador que determinan su estado en cada instante de tiempo de la ejecución:

- TCase: temperatura del procesador durante la ejecución.
- Velocidad del procesador durante la ejecución.
- Tiempo de variación de consumo: es el tiempo (cantidad en T unidades) para el cual se ha registrado el incremento de consumo energético del procesador. Cuando alcanza el valor indicado en el parámetro IV (Intervalo de Verificación) se determina si aplicar overclocking/underclocking.
- Variación de consumo: indica cuál es el incremento de consumo energético (en joules) para las últimas T unidades de tiempo (tiempo de variación de consumo).

Por cada instrucción ejecutada, el algoritmo de ajuste de rendimiento evalúa:

- Si la temperatura del procesador (TCase) es mayor a la temperatura máxima especificada (TCase Max):
    - Si el procesador soporta underclocking y la velocidad actual no es la mínima, se aplica underclock para disminuir la temperatura. Notar que si la velocidad es Min no es posible seguir disminuyendo su valor.
    - En caso contrario, el procesador se detiene hasta enfriar su temperatura a 25 grados y luego continúa su procesamiento. El rango de temperaturas para un procesador idle varía según la marca y modelo, pero generalmente corresponde a valores entre 25 y 35 grados.
- Si el incremento de consumo registrado durante el intervalo de verificación (IV), supera la variación de consumo esperada (VCE):
    - Si el procesador soporta underclocking y la velocidad actual no es la mínima, se aplica underclock para disminuir el consumo energético y la temperatura. Notar que si la velocidad es Min no es posible seguir disminuyendo su valor.
- Si el incremento de consumo registrado durante el intervalo de verificación (IV), es menor a la variación de consumo esperada (VCE):
    - Si el procesador soporta overclocking y la velocidad actual no es la máxima, se aplica overclock para incrementar la performance. Notar que si la velocidad es Max no es posible seguir aumentando la velocidad.

Las operaciones de overclock y underclock no se aplican mientras el tiempo de variación de consumo sea inferior al parámetro IV (Intervalo de Verificación). Cada vez que se aplica overclock, underclock o el robot se detiene para enfriarse, los valores de variación de consumo y tiempo transcurrido son reiniciados a 0.

5.4 Balance de Carga

ECMRE permite representar una arquitectura paralela heterogénea donde los cores trabajan a diferente velocidad, poseen distinto consumo energético y pueden utilizar o no técnicas de ajuste de rendimiento para incrementar la performance o normalizar los valores de consumo y temperatura cuando sea necesario. La existencia de cores con diferentes características es un factor fundamental que incide en el balance de carga. Como se explicó en el Capítulo 2 las tareas deben ser asignadas a los procesadores disponibles de forma equilibrada, intentando evitar que uno o más procesadores

estén ociosos esperando a que otros terminen su ejecución. Notar que el tiempo ocioso de uno o más procesadores, durante la ejecución, genera una disminución en los valores de las métricas como speedup y la eficiencia, presentadas en el capítulo 2.

El concepto de balance de carga puede ejemplificarse en ECMRE de forma sencilla. Por ejemplo, dados dos robots que recorren una avenida de principio a fin, pero tienen diferentes velocidades, es indudable que el más rápido quedará ocioso hasta que el otro termina, lo cual afecta al balance de carga.

El concepto de balance de carga incorpora una complejidad adicional en la resolución de problemas, pero también propone un nuevo desafío para el alumno a la hora de implementar un algoritmo que sea eficiente, lo que implica una motivación extra al utilizar el entorno ECMRE. El docente cuenta ahora con la oportunidad de elaborar enunciados donde el balance de carga sea uno de los factores a evaluar.

5.4.1 Balance de carga: ejemplo práctico

A continuación, se plantea un caso experimental sencillo para graficar el concepto de balance de carga y la mejora correspondiente del tiempo final de ejecución del algoritmo, a partir de una correcta asignación de tareas.

Problema 4: Cada robot debe recorrer una avenida completa y recoger todas las flores que encuentre en su camino. Se tienen 3 robot y cada uno recorre una avenida: Av1, Av2 y Av3. En cada avenida existen 25, 50 y 100 flores respectivamente. Los robots no soportan técnicas de ajuste de rendimiento y ejecutan a velocidad Max, Med y Min, respectivamente (Figura 5.15).

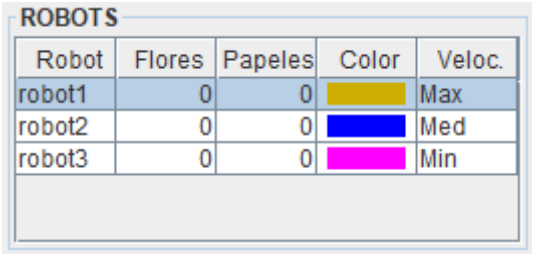


Figura 5.15: Problema 4 - configuración de velocidad.

Como se indica en las Tablas 5.1 y 5.2, el tiempo de la instrucción mover es 1T (Max), 2T (Med) y 4T (Min) y para tomarFlor es 2T (Max), 4T (Med) y 8T (Min).

Tiempo de ejecución (caso 1): los robots se asignan de la siguiente manera: el Robot1 (R1) recorre Av1, el Robot2 (R2) recorre Av2 y el Robot 3 (R3), la Av3. Como se observa en la Figura 5.16, el tiempo de ejecución por robot es:

- R1 en Max tarda 149 T
  - 99 mover → 99 * 1 T → 99 T
  - 25 tomarFlor → 25 * 2 T → 50 T
- R2 en Med tarda 398 T
  - 99 mover → 99 * 2 T → 198 T
  - 50 tomarFlor → 50 * 4 T → 200 T
- R3 en Min tarda 1196 T
  - 99 mover → 99 * 4 T → 396 T
  - 100 tomarFlor → 100 * 8 T → 800 T

El programa tiene una duración de 1196 T y el máximo tiempo ocioso es para R1 (1047 T).
El tiempo total ocioso (que es una medida del desbalance de carga) es de 1845 T (1047 + 798).

Figura 5.16: Resultado problema 4 - Tiempo de Ejecución - Caso 1: tiempo por robot.

Tiempo de ejecución (caso 2): los robots se asignan en forma inversa (R3 para Av1, R2 para Av2 y R1 para Av3). Como se observa en la Figura 5.17, el tiempo de ejecución por robot es:

- R1 en Min tarda 596 T
  - 99 mover → 99 * 4 T → 396 T
  - 25 tomarFlor → 25 * 8 T → 200 T
- R2 en Med tarda 398 T
  - 99 mover → 99 * 2 T → 198 T
  - 50 tomarFlor → 50 * 4 T → 200 T
- R3 en Max tarda 299 T
  - 99 mover → 99 * 1 T → 99 T
  - 100 tomarFlor → 100 * 2 T → 200 T

Notar que en este caso el programa tardaría 596 T y el máximo tiempo ocioso sería para R1 (297 T). El tiempo total ocioso desciende a 495 T.

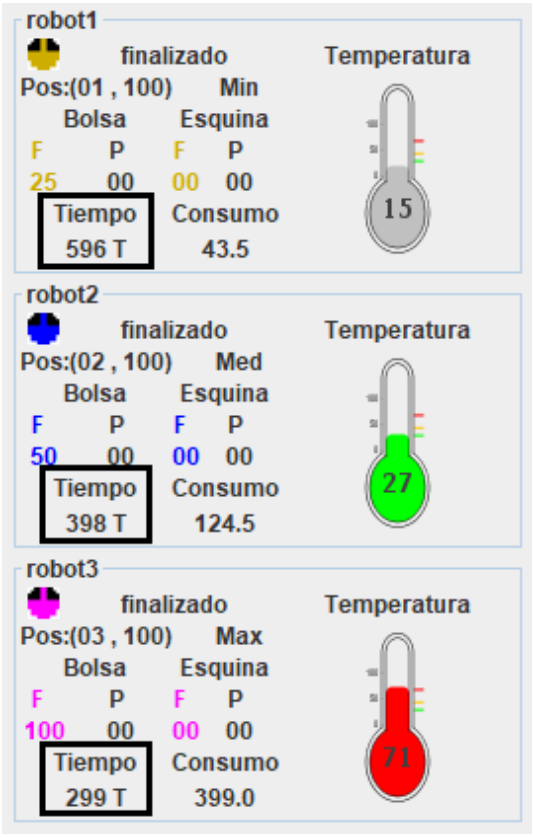


Figura 5.17: Resultado problema 4 - Tiempo de Ejecución - Caso 2: tiempo por robot.

El ejemplo anterior puede extenderse al caso de consumo. Dada una asignación de consumo energético de 1 Joule para la instrucción mover y 3 Joules para tomarFlor (ver Figura 5.18).

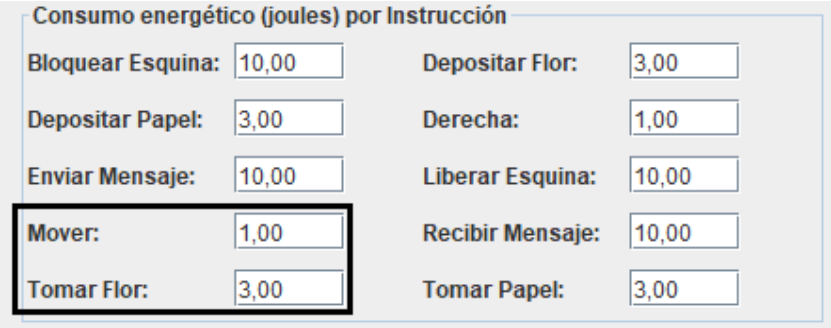


Figura 5.18: Problema 4 - configuración de consumo energético por instrucción.

Consumo (caso 1): los robots se asignan en forma directa (R1 para Av1, R2 para Av2 y R3 para Av3). Como se observa en la Figura 5.19, se obtiene el siguiente resultado:

- R1 en Max:
  - Consumo energético = (99 mover * 1 Joule * 1) + (25 tomarFlor * 3 Joules * 1) = 174 Joules
  - Tiempo de ejecución = (99 mover * 1 T) + (25 tomarFlor * 2 T) = 149 T
- R2 en Med:
  - Consumo energético = (99 mover * 1 Joule * 0.5) + (50 tomarFlor * 3 Joules * 0.5) = 124.5 Joules
  - Tiempo de ejecución = (99 mover * 2 T) + (50 tomarFlor * 4 T) = 398 T
- R3 en Min:
  - Consumo energético = (99 mover * 1 Joule * 0.25) + (100 tomarFlor * 3 Joules * 0.25) = 99.75 Joules
  - Tiempo de ejecución = (99 mover * 4 T) + (100 tomarFlor * 8 T) = 1196 T

El programa tarda 1196 T y consume en total 398.25 Joules.

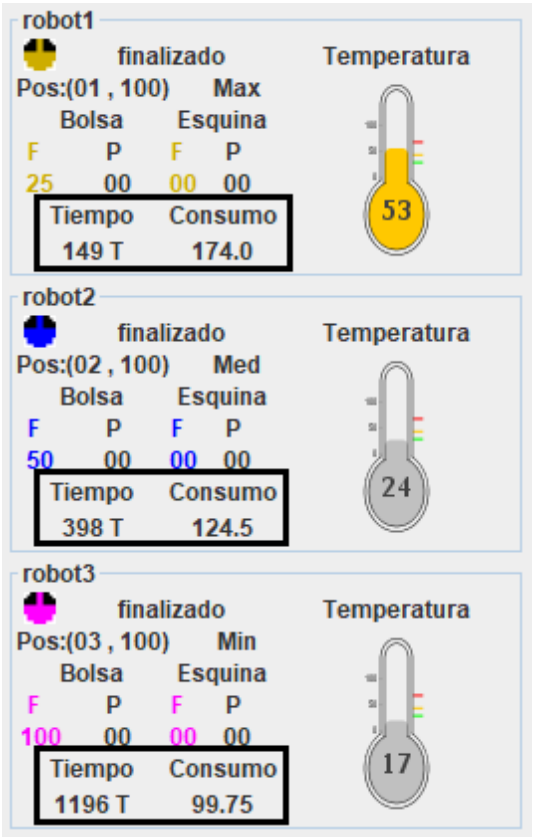


Figura 5.19: Resultado problema 4 - Consumo Energético - Caso 1: tiempo y consumo por robot.

Consumo (caso 2): los robots se asignan en forma inversa (R3 para Av1, R2 para Av2 y R1 para Av3). Como se observa en la Figura 5.20, se obtiene el siguiente resultado:

- R1 en Min recorre la Av1:
  - Consumo energético = (99 mover * 1 Joule * 0.25) + (25 tomarFlor * 3 Joules * 0.25) = 43.5 Joules
  - Tiempo de ejecución = (99 mover * 4 T) + (25 tomarFlor * 8 T) = 596 T
- R2 en Med recorre la Av2:
  - Consumo energético = (99 * 1 Joule * 0.5) + (50 * 3 Joules * 0.5) = 124.5 Joules
  - Tiempo de ejecución = (99 * 2 T) + (50 * 4 T) = 398 T
- R3 en Max recorre la Av3:
  - Consumo energético = (99 * 1 Joule * 1) + (100 * 3 Joules * 1) = 399 Joules
  - Tiempo de ejecución = (99 * 1 T) + (100 * 2 T) = 299 T

El programa tarda 596 T y consume un total de 567 Joules.

Figura 5.20: Resultado problema 4 - Consumo Energético - Caso 2: tiempo y consumo por robot.

Notar que a diferencia de lo ocurrido con el tiempo de ejecución la asignación de tareas realizada en el caso 2 tiene un mayor consumo energético. Cuando los procesadores ejecutan a mayor velocidad tienen un mayor consumo. La asignación planteada en el caso 2 tienen un mejor tiempo de ejecución y en consecuencia es esperable obtener un mayor consumo que en el caso 1. Dependerá del enunciado a resolver si se prioriza el tiempo de ejecución, el consumo energético o un equilibrio entre ambos.

5.5 Valores Globales del Sistema

Como se ha detallado anteriormente, ECMRE ha incorporado una serie valores constantes que son utilizados para el cálculo del tiempo de ejecución, consumo energético y temperatura, y por el algoritmo de ajuste de rendimiento. Estos valores constantes son:

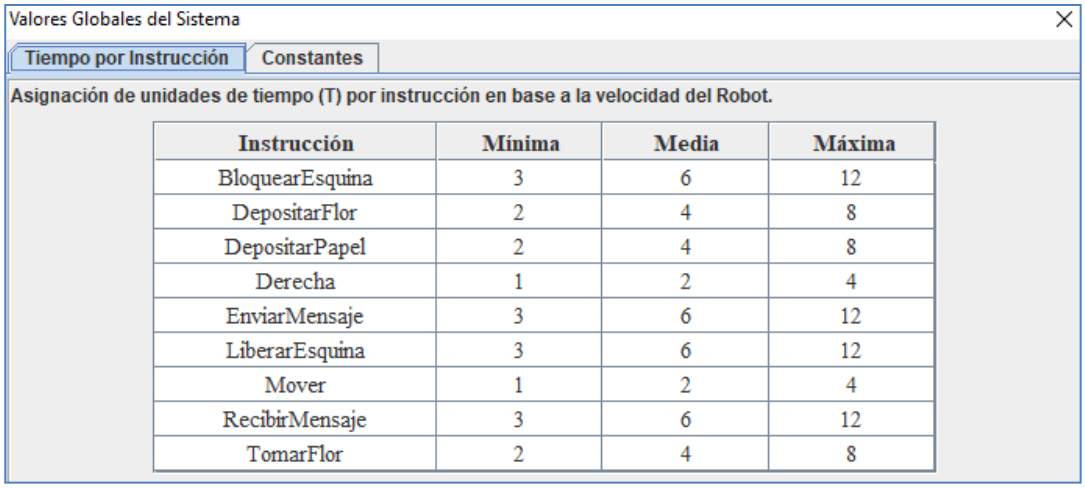
a) Cantidad de milisegundos que representa una unidad de tiempo T. Ver sección 5.2.1 Rendimiento de un procesador.
b) Cantidad de unidades de tiempo T asignadas a cada instrucción en base a la velocidad de ejecución. Ver sección 5.2.1 Rendimiento de un procesador.

c) Constantes de consumo por velocidad. Ver sección 5.2.2 Consumo de energía de un procesador.
d) Constantes de temperatura por velocidad. Ver sección 5.2.3 Temperatura de un procesador.
e) Valor de descenso de temperatura. Ver sección 5.3.1 Overclocking & Underclocking.

Se ha incorporado a ECMRE una nueva sección llamada Valores Globales del Sistema, la cual permite al alumno consultar los valores constantes detallados previamente. Los mismos se visualizan sólo a modo informativo y no pueden ser modificados por el usuario del sistema. La sección está compuesta por las pantallas que se presentan a continuación:

- Tiempo por Instrucción

  La primera pantalla (ver Figura 5.21) posee una tabla con la asignación de unidades de tiempo (T) por cada instrucción, en base a la velocidad de ejecución del robot. Esta asignación de tiempos fue presentada en las Tablas 5.1 y 5.2, y se incorpora a Valores Globales del Sistema posibilitando que el alumno consulte dicha información, la cuál será de utilidad para comprender cómo es calculado el tiempo durante la ejecución de su algoritmo.



Instrucción	Mínima	Media	Máxima
BloquearEsquina	3	6	12
DepositarFlor	2	4	8
DepositarPapel	2	4	8
Derecha	1	2	4
EnviarMensaje	3	6	12
LiberarEsquina	3	6	12
Mover	1	2	4
RecibirMensaje	3	6	12
TomarFlor	2	4	8

Figura 5.21: ECMRE - sección Valores Globales del Sistema: Tiempo por Instrucción.

- Constantes

  La segunda pantalla (ver Figura 5.22) muestra un conjunto de valores constantes utilizados para calcular el consumo energético y temperatura del procesador, y por el algoritmo de ajuste de rendimiento. Al igual que en la pantalla anterior, esta información puede ser consultada por el alumno.

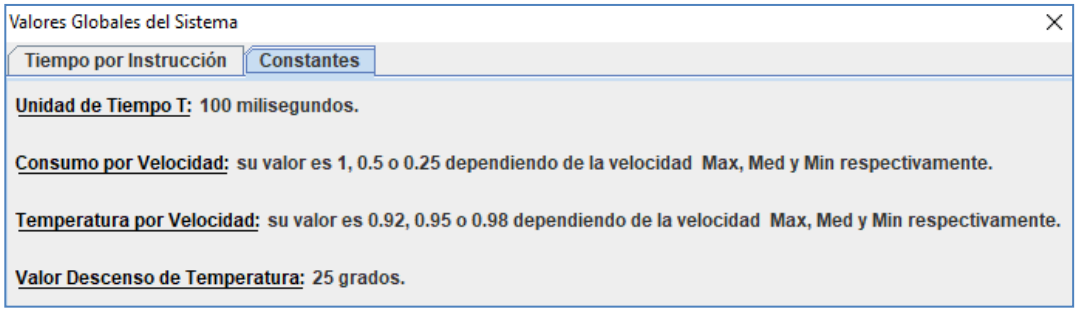


Figura 5.22: ECMRE - sección Valores Globales del Sistema: Constantes.

El usuario puede hacer click sobre el nombre de cada constante para ver una descripción de la misma. En la Figura 5.23 se visualiza el texto correspondiente a la constante Unidad de Tiempo T.

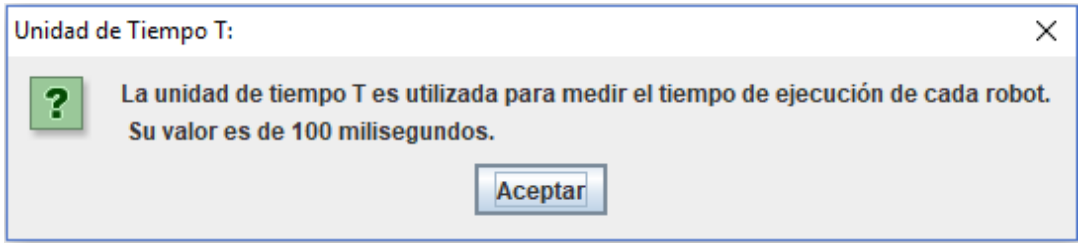


Figura 5.23: ECMRE - Descripción de la constante Unidad de Tiempo T.

5.6 Log del Procesador

Las nuevas funcionalidades incorporadas a ECMRE pueden requerir realizar un análisis post ejecución del algoritmo para evaluar si el resultado obtenido cumple con lo esperado. En el caso que el tiempo de ejecución o el consumo energético sea mayor al esperado, es posible que se proponga modificar la solución implementada o bien reasignar las tareas entre los procesadores involucrados para obtener un mejor resultado. Con el objetivo de facilitar este análisis se ha incorporado a ECMRE un nuevo módulo llamado Log del Procesador que muestra de forma amigable y resumida (mediante la utilización de tablas y gráficos) la ejecución de un algoritmo.

5.6.1 Logs

ECMRE registra información de los procesadores durante la ejecución del algoritmo. Dentro de la sección Robot / Procesador el usuario puede parametrizar cuál es la frecuencia de log durante la ejecución de cada robot (Figura 5.24). Al momento de compilar el algoritmo se asigna un valor, por defecto, que puede ser modificado por el alumno (antes de ejecutar su algoritmo) para permitir

ajustarlo a diferentes escenarios. Por ejemplo, si la frecuencia de log es 100, significa que se registrará una entrada de log cada 100 instrucciones ejecutadas. Las instrucciones que se contabilizan son las especificadas en la Tabla 5.1.

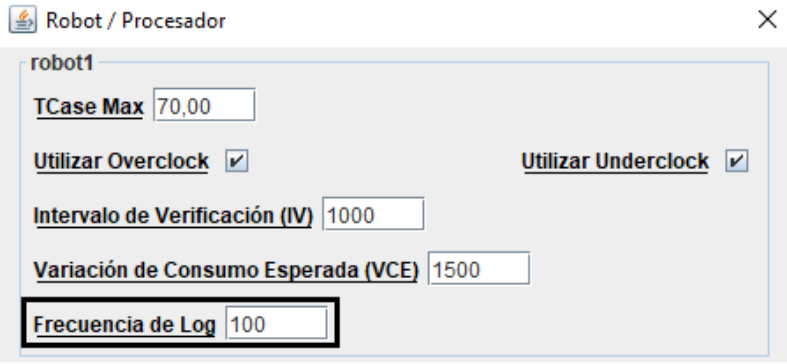


Figura 5.24: ECMRE - sección Robot / Procesador: Frecuencia de Log.

Independientemente de la frecuencia de log configurada hay un conjunto de eventos que son registrados siempre. Estos eventos pueden ser: procesador detenido por sobrecarga, overclock y underclock. Por cada entrada de log registrada se almacena la siguiente información del robot:

- Evento ocurrido: LOG, DETENIDO, OVERCLOCK y UNDERCLOCK.
- Velocidad antes de la ocurrencia del evento.
- Velocidad después de la ocurrencia del evento.
- Temperatura.
- Consumo energético.

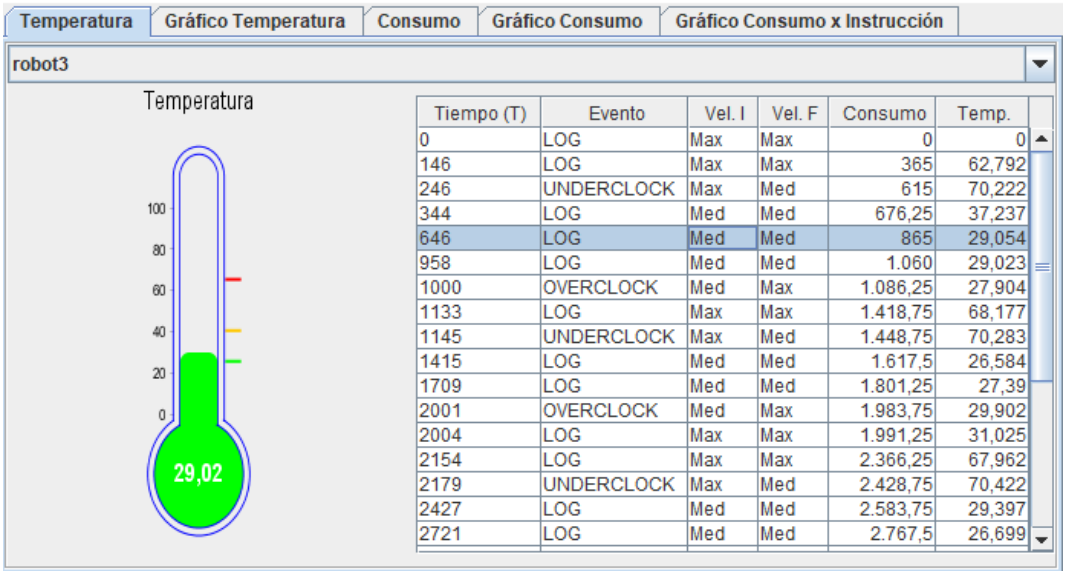
5.6.2 Tablas y gráficos

El nuevo módulo implementado en ECMRE y denominado Log del Procesador permite visualizar en forma de gráficos y tablas toda la información de log registrada durante la ejecución del algoritmo. El módulo cuenta con 5 secciones diferentes denominadas: Temperatura, Gráfico Temperatura, Consumo, Gráfico Consumo y Gráfico Consumo x Instrucción y en cada sección, el usuario puede seleccionar el robot, objeto de análisis.

- Temperatura

  La primera sección posee una tabla con las entradas de log registradas. Por ejemplo, en la Figura 5.25 se puede observar el intervalo de tiempo en el cual el procesador ha estado detenido, aplicado overclock o underclock. En la parte izquierda se presenta un termómetro

que muestra gráficamente la temperatura del procesador, cuando se selecciona la entrada de log de la tabla.



Tiempo (T)	Evento	Vel. I	Vel. F	Consumo	Temp.
0	LOG	Max	Max	0	0
146	LOG	Max	Max	365	62,792
246	UNDERCLOCK	Max	Med	615	70,222
344	LOG	Med	Med	676,25	37,237
646	LOG	Med	Med	865	29,054
958	LOG	Med	Med	1.060	29,023
1000	OVERCLOCK	Med	Max	1.086,25	27,904
1133	LOG	Max	Max	1.418,75	68,177
1145	UNDERCLOCK	Max	Med	1.448,75	70,283
1415	LOG	Med	Med	1.617,5	26,584
1709	LOG	Med	Med	1.801,25	27,39
2001	OVERCLOCK	Med	Max	1.983,75	29,902
2004	LOG	Max	Max	1.991,25	31,025
2154	LOG	Max	Max	2.366,25	67,962
2179	UNDERCLOCK	Max	Med	2.428,75	70,422
2427	LOG	Med	Med	2.583,75	29,397
2721	LOG	Med	Med	2.767,5	26,699

Figura 5.25: ECMRE - Log del Procesador: Temperatura

- Gráfico Temperatura

  La Figura 5.26 muestra mediante un gráfico de líneas, la variación de temperatura del procesador durante la ejecución.

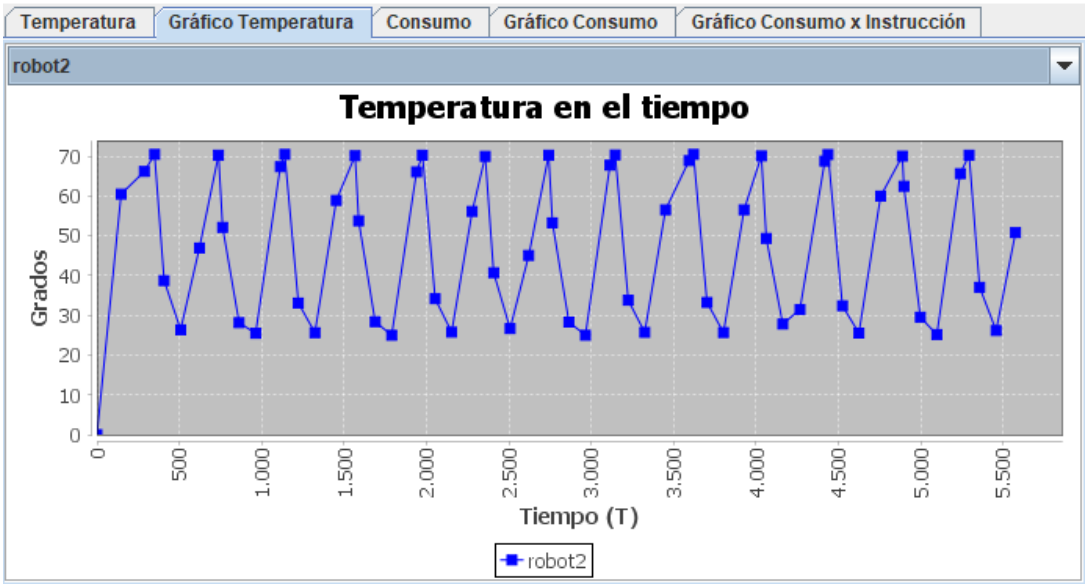
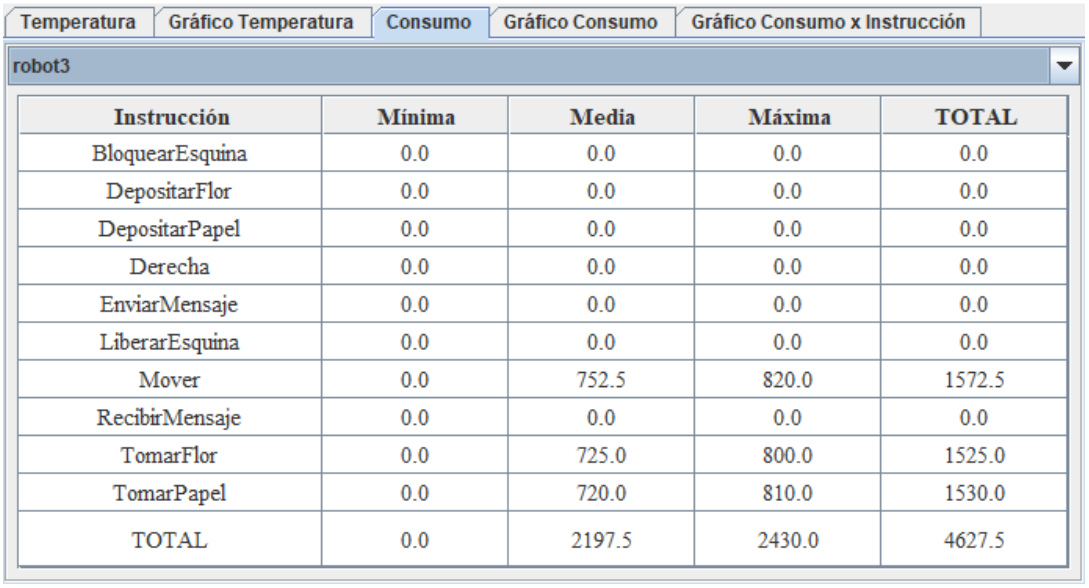


Figura 5.26: ECMRE - Log del Procesador: Gráfico Temperatura.

- Consumo

  En la Figura 5.27 se presenta la sección de consumo que permite visualizar mediante una tabla, el consumo correspondiente de velocidad, para cada instrucción.



Instrucción	Mínima	Media	Máxima	TOTAL
BloquearEsquina	0.0	0.0	0.0	0.0
DepositarFlor	0.0	0.0	0.0	0.0
DepositarPapel	0.0	0.0	0.0	0.0
Derecha	0.0	0.0	0.0	0.0
EnviarMensaje	0.0	0.0	0.0	0.0
LiberarEsquina	0.0	0.0	0.0	0.0
Mover	0.0	752.5	820.0	1572.5
RecibirMensaje	0.0	0.0	0.0	0.0
TomarFlor	0.0	725.0	800.0	1525.0
TomarPapel	0.0	720.0	810.0	1530.0
TOTAL	0.0	2197.5	2430.0	4627.5

Figura 5.27: ECMRE - Log del Procesador: Consumo.

- Gráfico Consumo

  La Figura 5.28, muestra mediante un gráfico de barras, el consumo del procesador durante la ejecución.

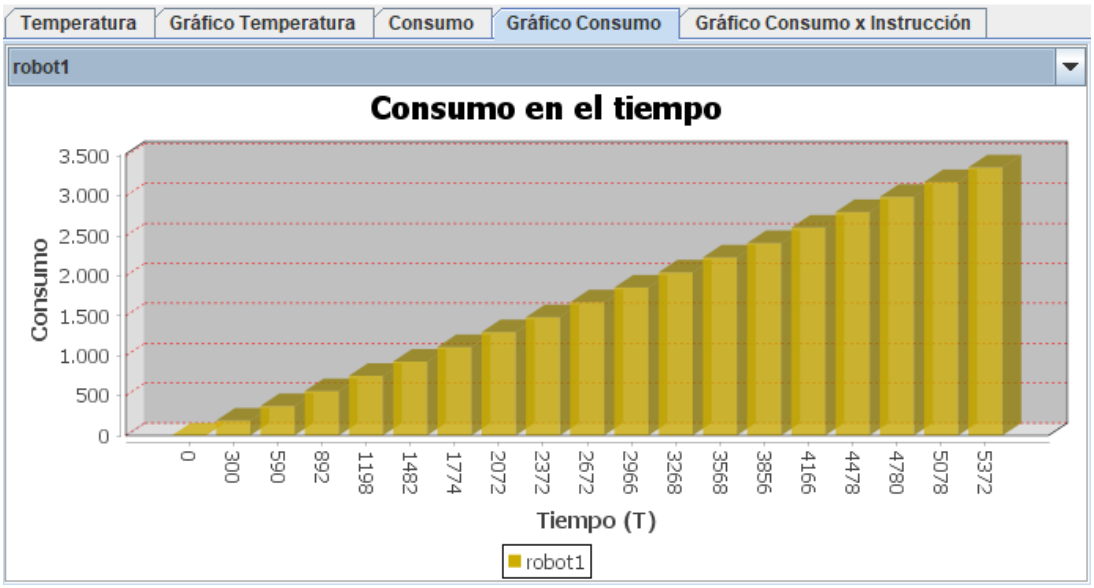


Figura 5.28: ECMRE - Log del Procesador: Gráfico Consumo.

- Gráfico Consumo x Instrucción

  La Figura 5.29 muestra mediante un gráfico de torta, el consumo del procesador por cada instrucción.

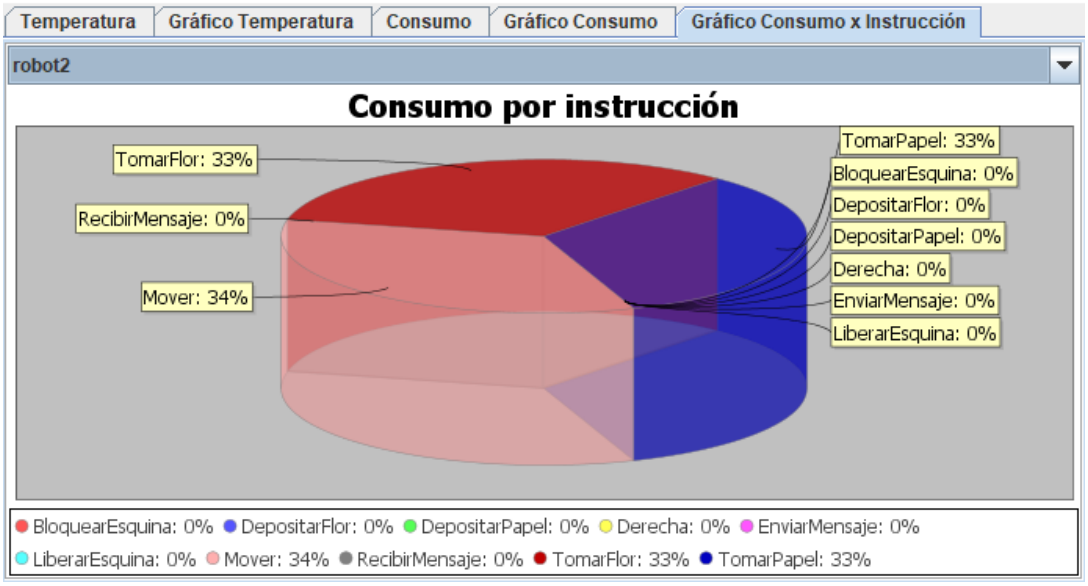


Figura 5.29: ECMRE - Log del Procesador: Gráfico Consumo x Instrucción

5.7 Resumen del Capítulo

En el capítulo 5 se ha realizado una descripción completa de la ampliación funcional realizada sobre el entorno CMRE (Concurrent Multi Robot Environment). Esta nueva versión del entorno ha sido denominada ECMRE (Extended Concurrent Multi Robot Environment) e incorpora los conceptos tratados en el capítulo 2, tales como: arquitecturas multicore homogéneas y heterogéneas, balance de carga, técnicas de ajuste de rendimiento, consumo energético y temperatura de los procesadores. Entre los cambios más relevantes realizados sobre el entorno se encuentra la posibilidad de ejecutar el algoritmo con robots que trabajan a distinta velocidad, la incorporación del tiempo de ejecución, consumo energético y temperatura por cada procesador/robot y la utilización de técnicas de ajustes como overclock y underclock, entre otras. En todo momento se ha priorizado la posibilidad de parametrizar estos nuevos conceptos, pudiendo activar o desactivar la funcionalidad por cada robot (sección Robot / Procesador). Se brinda al usuario una configuración por defecto que puede ser ajustada en base a sus necesidades concretas.

Es necesario resaltar que la definición del prototipo implementado ha priorizado la utilización de una interfaz gráfica y amigable que facilite los procesos de aprendizaje y enseñanza de los

conceptos involucrados. En el capítulo 3 se han mencionado los beneficios de una herramienta visual para abordar conceptos complejos. Por este motivo, se ha decidido que el tiempo de ejecución, consumo energético, temperatura y estado del procesador se visualicen en pantalla durante la ejecución de un algoritmo. La temperatura del procesador se representa mediante la imagen de un termómetro con valores indicados en grados que se relacionan con colores representativos. Por último, el módulo log del procesador brinda información de la ejecución en forma detallada a través de gráficos y tablas con el propósito de facilitar el proceso de análisis de los resultados obtenidos.

CAPITULO 6
EVALUACION Y RESULTADOS OBTENIDOS

6.1. Introducción

En el capítulo 5 se ha presentado el entorno ECMRE, el cuál incorpora conceptos informáticos básicos relacionados con la concurrencia y el paralelismo. Esta ampliación funcional contempla la utilización de múltiples robots/procesadores que trabajan en conjunto representando una arquitectura multicore. Cada robot tiene una velocidad, consumo energético y temperatura que son actualizados y visualizados durante la ejecución del algoritmo. Los robots admiten la utilización de técnicas de ajuste de rendimiento en base a la temperatura y el consumo del procesador. En el capítulo 1 se planteó la necesidad de presentarle a los alumnos de carreras de Informática, estos conceptos desde las asignaturas introductorias, principalmente porque las arquitecturas y los sistemas existentes que utilizarán durante su etapa como alumnos de esas carreras, son en esencia paralelas. La concurrencia, el paralelismo y los conceptos mencionados previamente son por naturaleza complejos. Desde el punto de vista educativo es un desafío abordar estos contenidos y contar con una herramienta interactiva, es algo sumamente útil.

La materia Taller de Programación de la Facultad de Informática de la UNLP corresponde al 1er año de las carreras y se compone de 3 módulos que plantean: a) el desarrollo de programas simples en el paradigma imperativo, b) el desarrollo de programas simples en un lenguaje orientado a objetos y c) a partir de la introducción de los conceptos básicos de la Programación Concurrente, el desarrollo de programas simples con un lenguaje de programación concurrente.

El módulo referido a Programación Concurrente es uno de los casos donde se utiliza el entorno CMRE para presentar los conceptos básicos de la concurrencia de modo que el alumno pueda visualizar de forma concreta conceptos informáticos que requieren un alto proceso de abstracción.

Se realiza una experiencia con alumnos del Taller de Programación, aprovechando el conocimiento y la práctica que tienen estos alumnos sobre el entorno CMRE. La experiencia tiene como propósito la realización de una actividad conjunta para trabajar los conceptos de temperatura, consumo y velocidad que incorpora ECMRE, plantear hipótesis de trabajo y contrastar con los resultados obtenidos, luego de la ejecución del programa. Al finalizar la actividad, se entregó una encuesta a los alumnos presentes con el propósito de obtener una devolución sobre los cambios realizados a CMRE.

6.2. Descripción de la sesión de prueba con alumnos

La sesión con alumnos se organiza en 3 momentos que se detallan en las subsecciones que siguen.

6.2.1 Presentación del entorno ECMRE

El entorno ECMRE fue presentado en una clase del módulo Programación Concurrente del Taller de Programación, durante el 1er cuatrimestre de 2017. Como ya se ha mencionado, los alumnos ya han utilizado el entorno CMRE para realizar las prácticas de dicho módulo.

La sesión se inició con un breve repaso de los conceptos de concurrencia y paralelismo abordados en el entorno:

- Arquitecturas Multicore
  - Homogéneas
  - Heterogéneas
- Procesadores
  - Consumo Energético
  - Temperatura
  - Técnicas de ajuste de rendimiento
- Balance de Carga

Para cada ítem se revisó el concepto y su importancia en relación a las arquitecturas de computadoras actuales y se presentaron los nuevos elementos (temperatura, consumo y velocidad) que han sido incorporados al entorno CMRE que dieron origen al nuevo entorno ECMRE que aborda dichos conceptos.

6.2.2. Planteo y resolución de la actividad práctica utilizando ECMRE

Una vez finalizada la revisión teórica, se planteó a los alumnos una actividad práctica que fue desarrollada con la supervisión del docente (tesista) y el conjunto de los alumnos. Los alumnos no interactuaron directamente con el entorno debido a que aún estaba en la etapa de desarrollo. La actividad práctica consistió en la resolución del siguiente problema, utilizando el entorno ECMRE:

Escriba un programa para recorrer las 3 áreas de la figura 6.1 limpiando las esquinas de flores y papeles. Cada área es un cuadrado de 30 x 30 en el que se encuentran 450 flores y 450 papeles.

Se utilizarán 3 robots (uno por cada área) y se desea obtener un resultado eficiente en tiempo de ejecución y consumo energético.

El límite máximo de temperatura (Tcase Max) es de 70 grados para cada uno de los robots. Se permite elegir la velocidad de los procesadores, y el uso de las técnicas de ajuste de Overclocking y Underclocking.

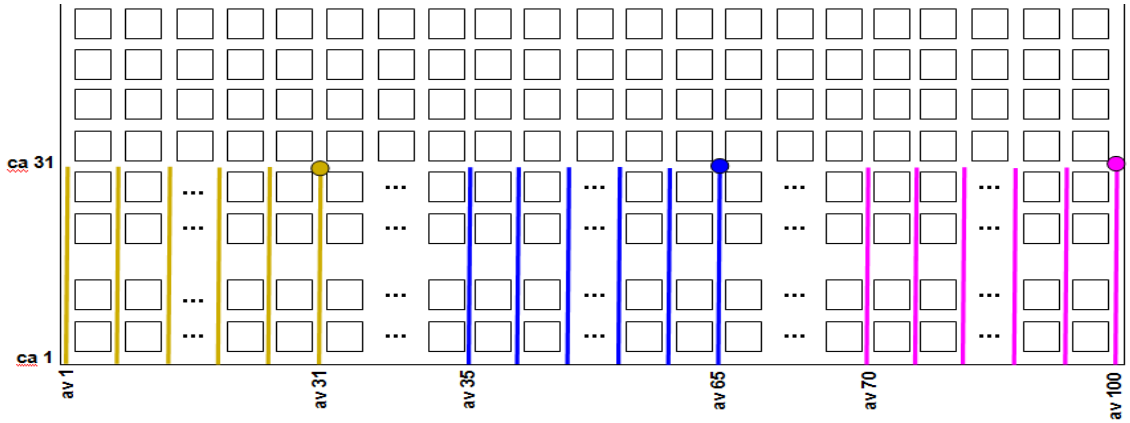


Figura 6.1: Recorrido ejercicio práctico.

El docente establece que la configuración de consumo (joules) y la temperatura (grados) por instrucción utilizada, es la misma en todos los procesadores (ver Figura 6.2):

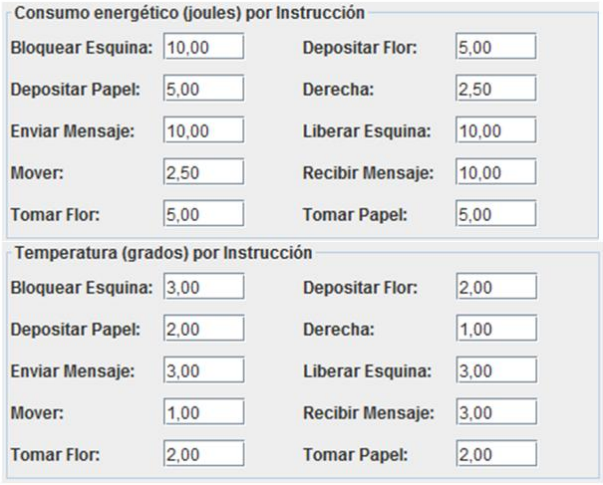


Figura 6.2: ECMRE - Consumo energético y temperatura por instrucción en sección Robot/Procesador.

El docente recuerda que como indica el enunciado, es posible seleccionar la velocidad de los procesadores y configurar si hacen uso o no de las técnicas de ajuste como Overclocking y Underclocking. Se decide en conjunto con los alumnos, utilizar 3 robots/procesadores con

diferentes características (ver Tabla 6.1) para evaluar luego los resultados obtenidos y determinar aquel que ofrece un mejor resultado, de acuerdo a lo que pide el enunciado del problema.

	Robot 1	**Robot2**	**Robot3**
Velocidad	Med	Max	Max
Overclocking	NO	NO	SI
Underclocking	NO	NO	SI

Tabla 6.1: Características por robot a utilizar en ejercicio práctico.

El docente remarcó que todos los robots tienen la misma carga de trabajo (realizan la misma tarea). También aclaró que cada robot debía recorrer un cuadrado de 30 x 30 recogiendo 450 flores y 450 papeles. Antes de pasar a la visualización de la ejecución del programa, el docente y los alumnos analizaron los resultados esperados en base a la configuración de cada robot y plantearon las siguientes hipótesis:

1. Aquellos robots que tienen velocidad Max deberían obtener una mejor performance en cuanto a tiempo de ejecución.
2. Los robots 2 y 3 trabajan a velocidad Max, sin embargo, el robot 3 soporta underclocking y puede disminuir su velocidad, de acuerdo al incremento de temperatura. Se puede asumir que el robot 2 será el que tendrá un mejor rendimiento en cuanto a tiempo ya que siempre ejecutará a velocidad Max.
3. El robot 1 debería tener un consumo menor que el resto ya que trabaja a menor velocidad.

Una vez definidas las hipótesis mencionadas, se prosiguió con la ejecución del algoritmo observando y analizando la actualización del tiempo, el consumo y la temperatura y a la vez, identificando las técnicas de rendimiento utilizadas. El resultado final obtenido fue el siguiente (ver Figura 6.3):

- Tiempo de Ejecución
    1. robot 3 (4502 T)
    2. robot 1 (5460 T)
    3. robot 2 (5866 T)
- Consumo Energético
    1. robot 1 (3412.5 Joules)
    2. robot 3 (4610 Joules)
    3. robot 2 (6825 Joules)

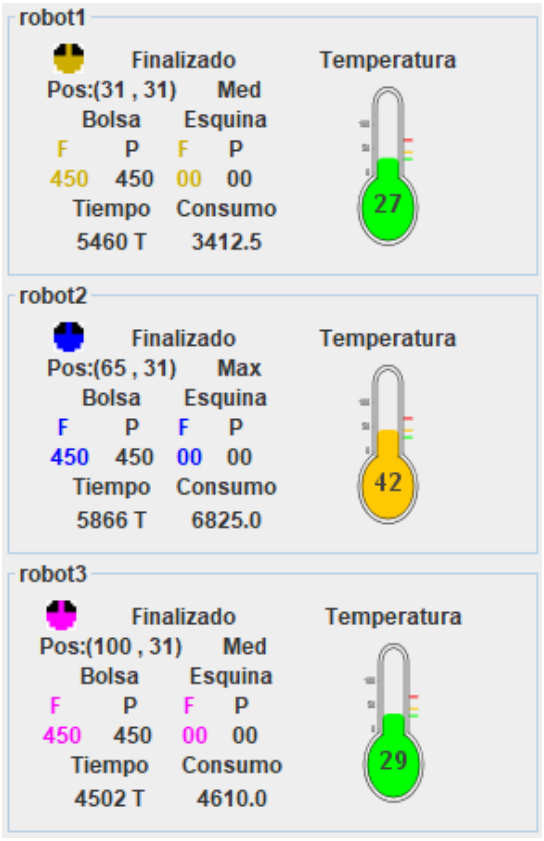


Figura 6.3 - ECMRE: resultado ejecución ejercicio práctico.

El docente propuso utilizar el módulo Log del Procesador para acceder a los gráficos que permiten observar la variación de la temperatura y el consumo durante la ejecución del algoritmo. En las Figuras 6.4, 6.5 y 6.6 se muestra la variación de temperatura por cada robot durante la ejecución del algoritmo:

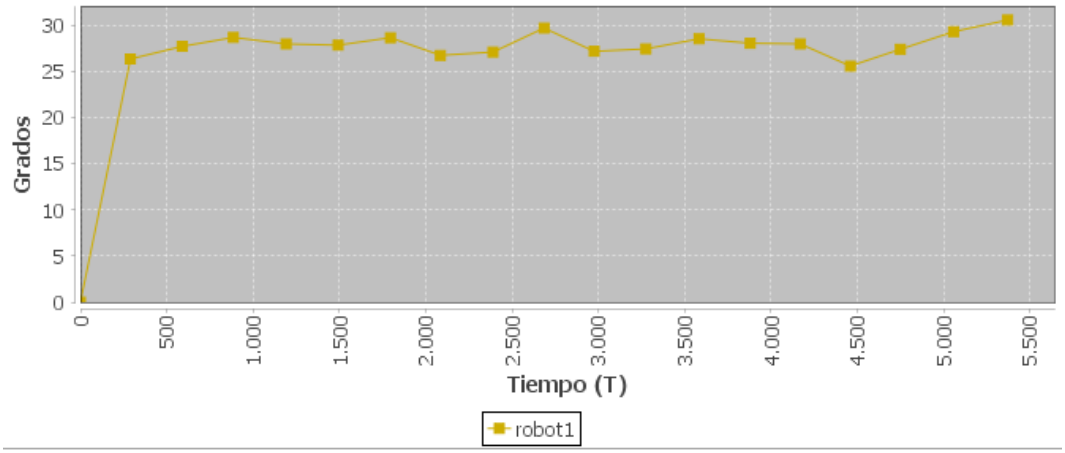


Figura 6.4: ECMRE - gráfico que indica la temperatura, en relación al tiempo empleado por el robot1.

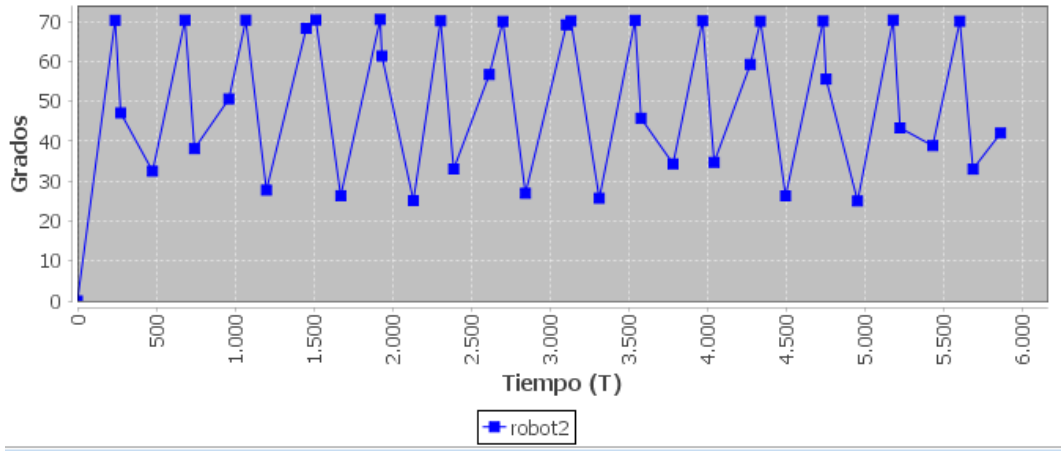


Figura 6.5: ECMRE - gráfico que indica la temperatura, en relación al tiempo empleado por el robot2.

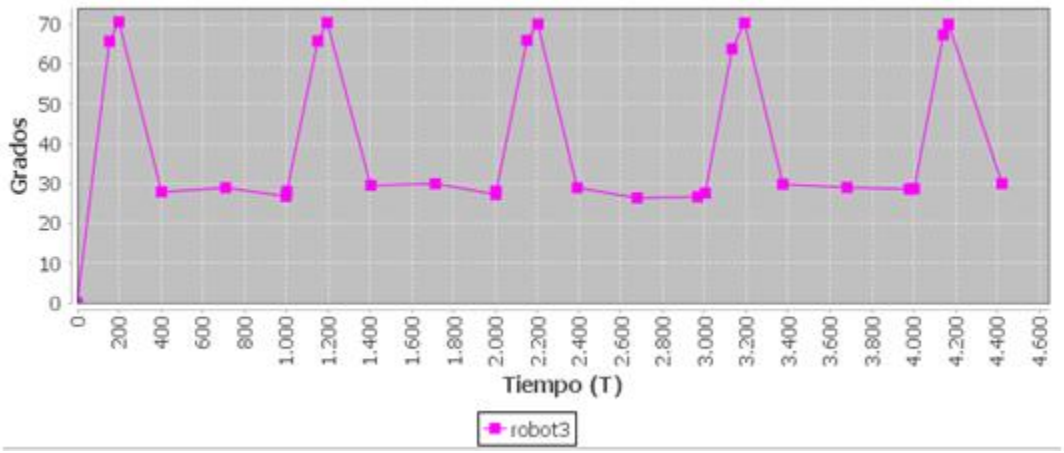


Figura 6.6: ECMRE - gráfico que indica la temperatura, en relación al tiempo empleado por el robot3.

En las Figuras 6.7, 6.8 y 6.9 se muestra el incremento de consumo energético por cada robot durante la ejecución del algoritmo:

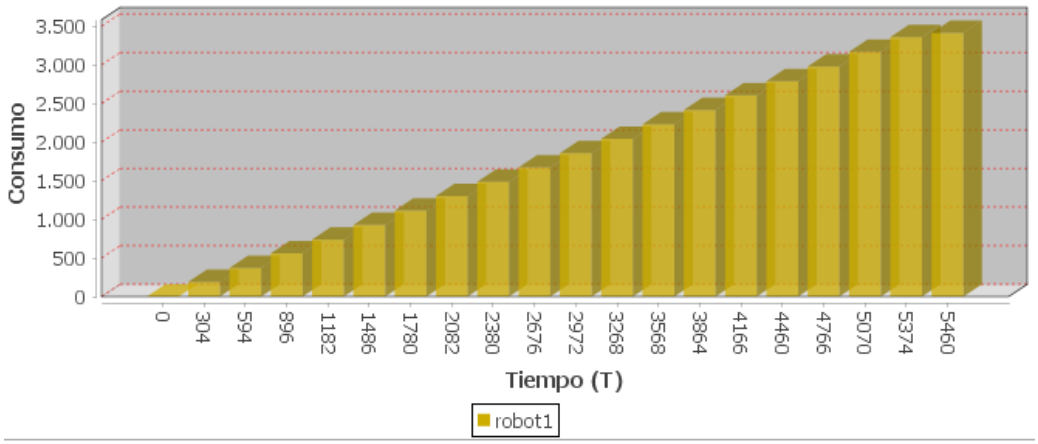


Figura 6.7: ECMRE - consumo energético, en relación al tiempo para el robot1.

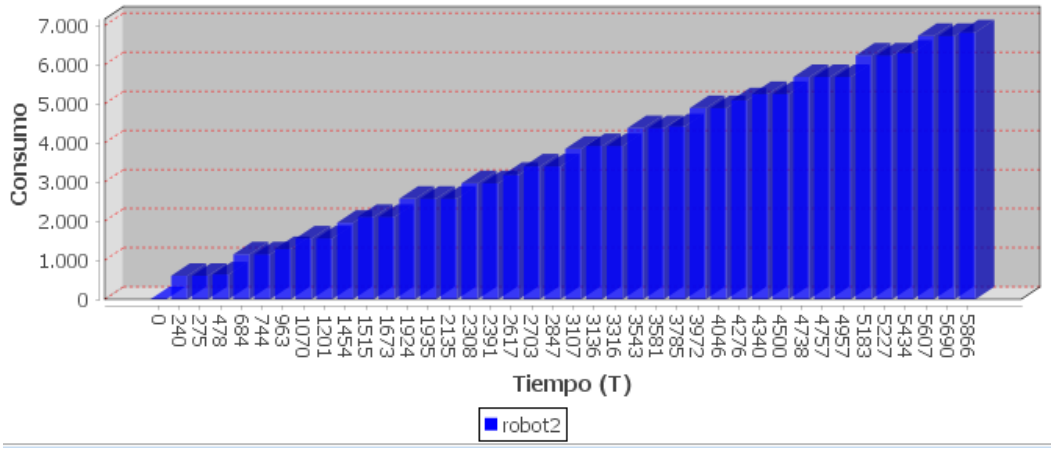


Figura 6.8: ECMRE - consumo energético, en relación al tiempo para el robot2.

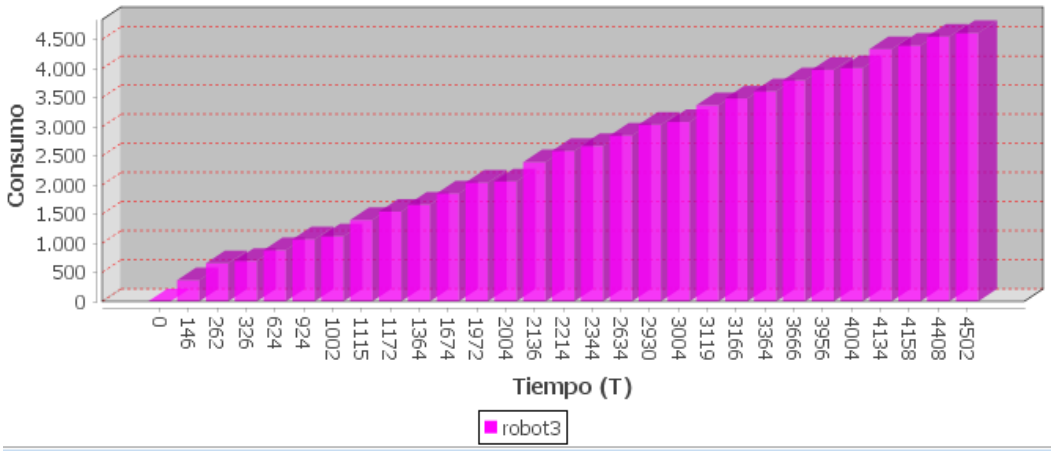


Figura 6.9: ECMRE: consumo energético, en relación al tiempo para el robot3.

Una vez finalizada la ejecución del programa, el docente y los alumnos validaron las hipótesis planteadas previamente:

1. La primera hipótesis plantea que aquellos robots que trabajan a velocidad Max deberían obtener una mejor performance en cuánto al tiempo de ejecución.
   En base a los resultados obtenidos la hipótesis es **falsa**. El robot 2 con velocidad Max es el que ha tenido una peor performance. Esto se debe a que ha superado su máxima temperatura en reiteradas ocasiones debiendo detenerse para enfriarse.
2. La segunda hipótesis espera que el robot 2 tenga un mayor rendimiento ya que siempre ejecuta a velocidad Max. Aunque el robot 3 también trabaja a velocidad Max, soporta underclocking y puede disminuir su velocidad, de acuerdo al incremento de temperatura y consumo energético.
   En base a los resultados obtenidos la hipótesis es **falsa**. El robot 3 que soporta underclocking ha tenido un mejor rendimiento ya que el robot 2 ha colapsado en varias oportunidades, debiendo detenerse para disminuir la temperatura.
3. La tercera hipótesis indica que el robot 1 debía tener un consumo menor que el resto ya que trabaja a menor velocidad.
   En base a los resultados obtenidos la hipótesis es **verdadera**.

De acuerdo a los resultados obtenidos de los gráficos, el análisis realizado luego de la ejecución del programa y la validación de las hipótesis formuladas, en conjunto con los alumnos se concluyó que:

- ✓ La utilización de las técnicas de ajuste (overclocking y underclocking) mejora notablemente el rendimiento de los robots/procesador cuando el algoritmo ejecutado provoca el incremento de sus temperaturas superando el límite máximo (TCase Max).
- ✓ Si se busca tener un ahorro en el consumo, la utilización de un procesador a velocidad media ofrece buenos resultados con un tiempo de ejecución aceptable.
- ✓ Si se quiere priorizar el tiempo de ejecución por sobre el consumo energético, la utilización de un procesador a máxima velocidad que soporte overclocking y underclocking puede resultar la mejor opción.

6.2.3 Encuesta final

Al finalizar la actividad práctica, se entregó a cada alumno una breve encuesta, anónima, con el objetivo de tener un primer feedback por parte de los alumnos del uso de ECMRE, quienes serán usuarios finales de la herramienta. Cabe aclarar que dentro de los contenidos específicos de la

asignatura Taller de Programación no se incluyen actualmente los conceptos incorporados en ECMRE, y que los alumnos encuestados no trabajaron individualmente con la nueva versión del entorno. Sin embargo, dado que la asignatura sí utiliza la versión CMRE para introducir conceptos básicos de concurrencia, se consideró de interés contar con una experiencia de uso con los alumnos de Taller de Programación, para obtener un feedback sobre la nueva versión del entorno.

La encuesta realizada se encuentra en el Anexo I y se compone de 5 preguntas cuyas respuestas cubren una escala de Likert que varía de 1 (en completo desacuerdo) a 5 (completamente de acuerdo). Si el alumno no se siente capaz de responder a un determinado ítem, puede elegir el valor del centro en la escala (3). Como ya se mencionó, esta encuesta es anónima para que el alumno pueda responder sin condicionamientos. La encuesta se realizó a los 42 alumnos que asistieron a la clase de Taller, el día de la experiencia y se obtuvieron los siguientes resultados:

1. **La aplicación ECMRE colabora en el aprendizaje de los conceptos presentados.**

   En base a los contenidos teóricos presentados en la clase (temperatura, consumo energético, técnicas de ajuste de rendimiento, entre otras) y el ejercicio práctico realizado en ECMRE, se consulta al alumno si cree que las nuevas funcionalidades incorporadas al entorno resultarían útiles para aprender estos nuevos conceptos. El 93% de los alumnos encuestados (36% completamente de acuerdo y 57% de acuerdo) cree que la aplicación puede colaborar en el proceso de aprendizaje. El 7% restante respondió que no está ni de acuerdo, ni en desacuerdo con la afirmación realizada (Figura 6.10).

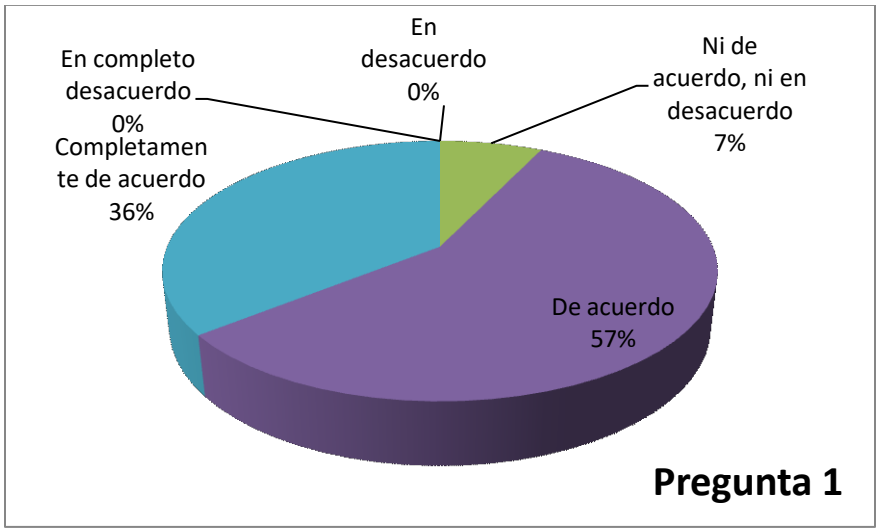


Figura 6.10: Encuesta - resultados pregunta 1.

2. **Es útil contar con una herramienta de aplicación práctica que permita mediante ejemplos concretos visualizar los contenidos teóricos aprendidos en clase.**

La segunda pregunta intenta medir cual es la percepción del alumno sobre la utilidad de la herramienta para aprender los conceptos teóricos. El 93% de los alumnos encuestados (55% completamente de acuerdo y 38% de acuerdo) cree que es útil contar con una herramienta de aplicación práctica para visualizar los contenidos teóricos. El 7% restante respondió que no está ni de acuerdo, ni en desacuerdo con la afirmación realizada (Figura 6.11).

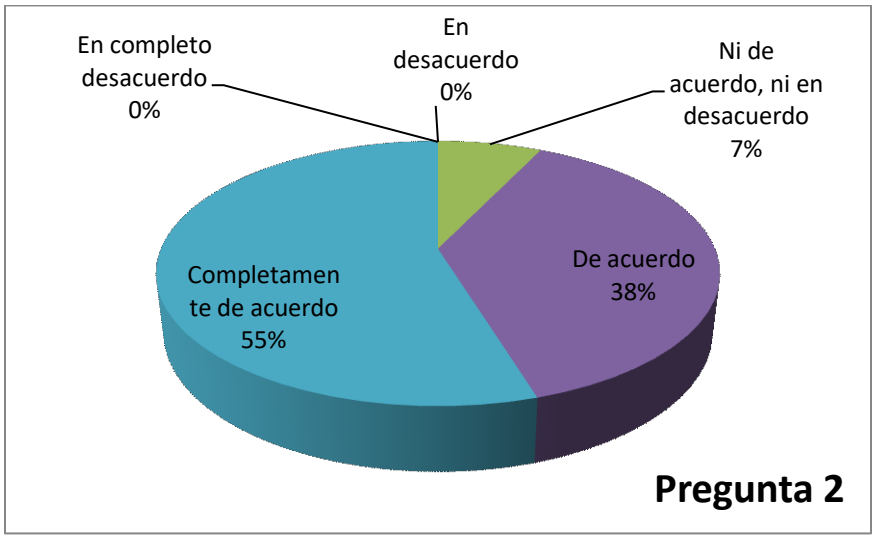


Figura 6.11: Encuesta - resultados pregunta 2.

3. **Los contenidos en ECMRE se encuentran organizados y su utilización es intuitiva.**

La tercera pregunta hace referencia a la usabilidad del sistema. El 76% de los alumnos encuestados (52% completamente de acuerdo y 24% de acuerdo) cree que la aplicación tiene una organización adecuada e intuitiva de los contenidos. El 24% restante respondió que no está ni de acuerdo, ni en desacuerdo con la afirmación realizada (Figura 6.12).

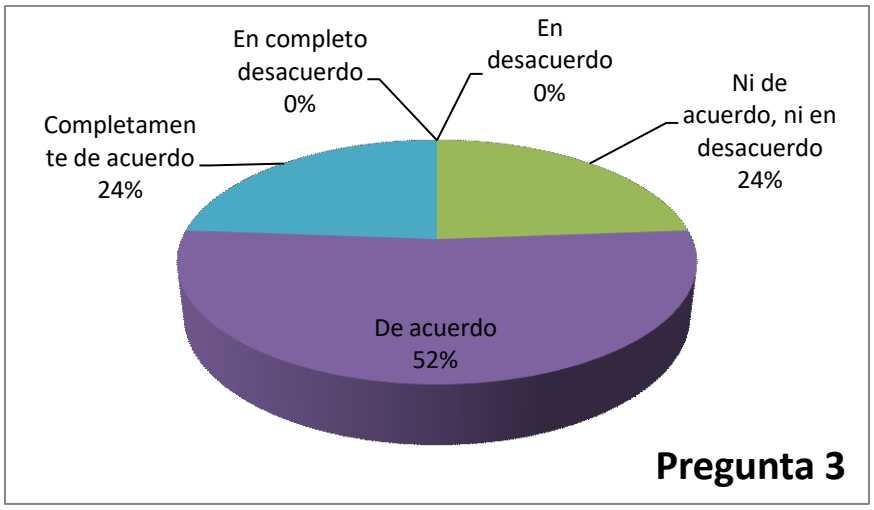


Figura 6.12: Encuesta - resultados pregunta 3.

4. **La iconografía y gráficos utilizados en la aplicación tienen el tamaño adecuado y coinciden con la función asociada.**

   Al igual que en la pregunta anterior se consulta al alumno sobre aspectos relacionados con la interface de ECMRE. En particular aquí se analizan los íconos y gráficos incorporados. Notar por ejemplo la utilización de un termómetro con colores que representan la temperatura del procesador en ejecución. El 86% de los alumnos encuestados (48% completamente de acuerdo y 38% de acuerdo) cree que los íconos y gráficos son adecuados. El 14% restante respondió que no está ni de acuerdo, ni en desacuerdo con la afirmación realizada (Figura 6.13).

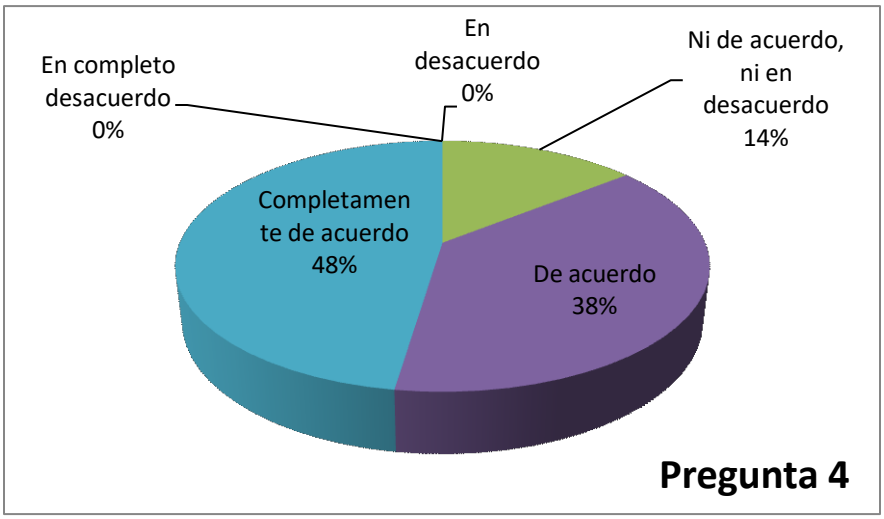


Figura 6.13: Encuesta - resultados pregunta 4.

5. **Es beneficioso contar con esta aplicación como complemento a las clases teóricas.**

   Por último, se consulta a los alumnos si creen que contar con ECMRE como un recurso complementario a las clases teóricas podría ser beneficioso. El 88% de los alumnos encuestados (52% completamente de acuerdo y 36% de acuerdo) coincide en que la utilización de la aplicación como complemento a la teoría colabora con el aprendizaje de los temas vistos. El alto porcentaje obtenido puede estar relacionado con el interés de los alumnos en aprender mediante ejemplos prácticos en los que puede participar de forma activa. El 12% restante respondió que no está ni de acuerdo, ni en desacuerdo con la afirmación realizada (Figura 6.14).

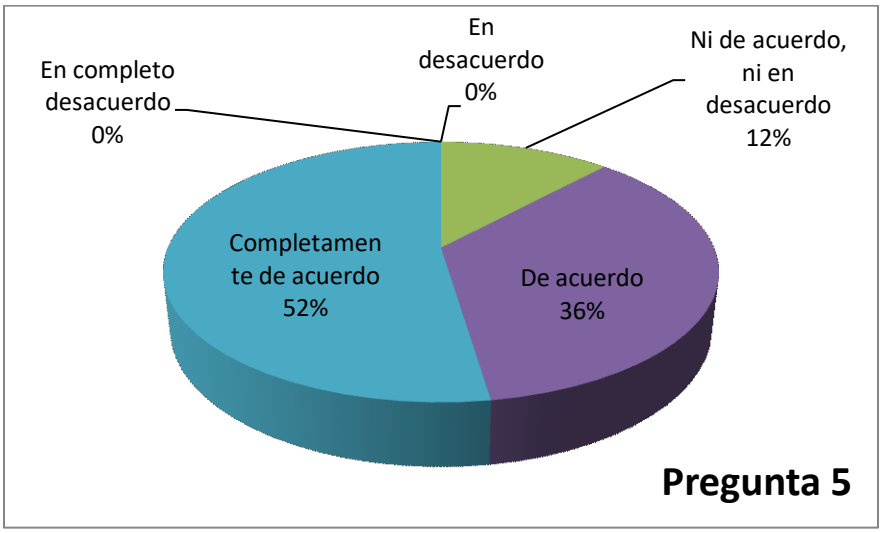


Figura 6.14: Encuesta - resultados pregunta 5.

Como se observa en el Anexo I, cada pregunta de la encuesta tiene una sección de observaciones donde es posible dejar algún comentario adicional. Las observaciones realizadas por los alumnos se pueden agrupar en 3 categorías:

1. Remarcan la utilidad de llevar a la práctica los contenidos vistos en teoría.
2. Indican su interés por utilizar el entorno ECMRE en caso de incorporar estos nuevos contenidos a la materia.
3. Señalan que al tener experiencia en el uso del entorno en su versión anterior (CMRE), pueden proponer mejoras relacionadas con la usabilidad del entorno CMRE. En especial se sugieren mejoras para facilitar el uso del editor de código. También se propone alguna función de guardado (del estado de la aplicación) para ejecutar sucesivas veces el mismo escenario, sin la necesidad de volver a configurar el estado inicial. Si bien estas

observaciones se proponen desde la perspectiva de uso de CMRE, podrán ser consideradas como mejoras del entorno ECMRE.

6.3. Resumen del Capítulo

En el capítulo 6 se describe y analiza la presentación del entorno ECMRE realizada durante una clase del módulo Programación Concurrente del Taller de Programación de la Facultad de Informática de la UNLP. Se repasan los contenidos teóricos vistos por los alumnos y se presentan otros relacionados (temperatura, consumo y velocidad) a partir de la implementación de un programa en ECMRE, que resuelve una actividad práctica planteada. Por último, se hace una evaluación de los resultados obtenidos a partir de la realización de una encuesta a los alumnos presentes en la clase.

CAPITULO 7
CONCLUSIONES

Uno de los principales desafíos que enfrentan los docentes universitarios en la enseñanza de carreras informáticas, es la necesidad de mantener una currícula vigente y moderna. En esta tesina de grado se analiza la necesidad de presentar al alumno, ciertos conceptos de concurrencia y paralelismo que han tomado mayor relevancia en los últimos años y son parte en la actualidad de cualquier arquitectura de computadora. Con el objetivo de encontrar estrategias y herramientas que hagan posible abordar de forma temprana estos conceptos fundamentales de concurrencia y paralelismo que son por su naturaleza complejos, se ha definido e implementado una extensión del entorno CMRE llamada ECMRE (Extended Concurrent Multi Robot Environment).

ECMRE es una herramienta interactiva para la enseñanza de conceptos de concurrencia y paralelismo. Permite visualizar de forma gráfica (mediante la utilización de ejemplos concretos) estos contenidos que requieren un alto grado de abstracción en el proceso de aprendizaje. Se espera que la implementación realizada, colabore en el proceso de enseñanza y aprendizaje de estos temas en los cursos iniciales de programación, así como también en cursos de años superiores.

Revisando los objetivos planteados en la propuesta del trabajo, se enumeran las conclusiones a los mismos:

Objetivos Generales

- *Incorporar al entorno CMRE algunos conceptos relacionados a los ya existentes de Concurrencia y Paralelismo para su utilización en asignaturas de los primeros años de Informática.*

Se pude concluir que el objetivo general planteado inicialmente en la propuesta de trabajo se ha cumplido exitosamente. Todos los contenidos de Concurrencia y Paralelismo investigados en el capítulo 2, pueden verse reflejados en la implementación del entorno ECMRE especificada en el capítulo 5. La definición del prototipo implementado ha priorizado la utilización de una interfaz gráfica y amigable que facilite los procesos de aprendizaje y enseñanza en las asignaturas de los primeros años de Informática.

Objetivos Específicos

- *Investigar conceptos de procesadores multicore y sus arquitecturas, como el consumo de energía, temperatura de los procesadores, balance de carga de la aplicación, entre otros.*

En el capítulo 2 se trabajó en su totalidad sobre este objetivo. Los procesadores multicore fueron el eje principal de investigación; se clasificaron las arquitecturas multicore, se analizó su rendimiento y se focalizo en dos problemáticas actuales como son el incremento de consumo energético y la temperatura generada por dichos procesadores. Se describen técnicas y tecnologías utilizadas por los diseñadores de arquitecturas multicore para atacar estas problemáticas. Desde el punto de vista de la programación, se analiza cómo ha sido afectada por los procesadores multicore y la necesidad de crear aplicaciones que exploten al máximo todos sus núcleos (balanceando la carga de trabajo) y al mismo tiempo sean eficientes desde el punto de vista energético.

- Extender el entorno CMRE (Concurrent Multi Robot Environment) permitiendo la posibilidad de variar las velocidades de los múltiples robots. Esto se verá reflejado gráficamente a través del consumo y la temperatura de los mismos. Además, se permitirá acelerar/disminuir la velocidad de los robots (potencia de cómputo) en base al estado (temperatura) de los mismos durante la ejecución.

En el capítulo 5 se ha realizado una descripción completa de la ampliación funcional realizada sobre el entorno CMRE (Concurrent Multi Robot Environment). Esta nueva versión del entorno denominada ECMRE (Extended Concurrent Multi Robot Environment) incorpora los conceptos tratados en el capítulo 2. El entorno permite ejecutar un algoritmo con robots que trabajan a distinta velocidad, e incorpora tiempo de ejecución, consumo energético y temperatura por cada procesador/robot. En base a los valores de consumo energético y temperatura de cada robot durante la ejecución, es posible aplicar técnicas de ajustes de rendimiento como overclock y underclock. La implementación realizada ha priorizado la posibilidad de parametrizar estos nuevos conceptos, pudiendo activar o desactivar la funcionalidad por cada robot (sección Robot / Procesador). Se brinda al usuario una configuración por defecto que puede ser ajustada en base a sus necesidades concretas.

En el capítulo 3 se plantean las ventajas de contar con una herramienta visual e interactiva para enseñar contenidos complejos. Por esta razón en ECMRE se actualiza y visualiza el tiempo, consumo energético, temperatura y estado del procesador durante la ejecución. La temperatura del procesador es graficada mediante la utilización de un termómetro con su valor numérico en grados y colores representativos. Una vez finalizada la ejecución, será necesario realizar un proceso de análisis para determinar si el algoritmo es eficiente desde el punto de vista de tiempo y consumo energético. Para facilitar esta tarea, se ha implementado un nuevo módulo de log del procesador que brinda información de la ejecución en forma detallada, a través de gráficos y tablas.

- Presentar los conceptos incorporados al entorno CMRE en las asignaturas de los primeros años de las carreras de la Facultad de Informática de la UNLP. Generar un informe en base a los resultados obtenidos.

En el capítulo 6 se describe y analiza la presentación del entorno ECMRE realizada en la materia Taller de Programación de la Facultad de Informática de la UNLP. Se trabajó con un grupo de 42 alumnos durante el transcurso de una clase en la que se presentaron los contenidos teóricos incorporados al entorno y se trabajó con la herramienta ECMRE, mediante la implementación de un ejercicio práctico. Por último, se realizó una encuesta a los alumnos obteniendo así un primer feedback de la aplicación que en base a los resultados obtenidos, deja un saldo positivo del trabajo realizado en la tesina.

TRABAJOS FUTUROS

La implementación del entorno ECMRE es una extensión funcional de la herramienta CMRE en su versión actual. En base al trabajo realizado con esta nueva versión del entorno, se proponen una serie de mejoras y líneas de trabajo e investigación a futuro para este proyecto:

- Publicar la aplicación ECMRE para que docentes y alumnos puedan descargar el ejecutable de esta nueva versión del entorno. Proveer una vía de comunicación para canalizar y atender posibles consultas en cuanto a la utilización del entorno, reportes de bugs, mejoras funcionales, entre otros. Facilitar el acceso a un manual de usuario que presente los cambios realizados en la aplicación, tal como se hizo en el capítulo 5 de esta tesina.
- ECMRE provee un alto grado de parametrización que permite al usuario representar distintos tipos de arquitecturas multicore. Por cada robot/procesador es posible configurar su temperatura máxima, si utiliza o no técnicas de ajuste de rendimiento, valores de consumo energético y temperatura por instrucción, entre otras. Sin embargo, existen algunos valores constantes presentados en el capítulo 5 que podrían ser editables para tener aún un control mayor sobre el comportamiento de los procesadores. Algunos ejemplos de estas constantes son el valor de la unidad de tiempo T (100 milisegundos), valor de descenso de temperatura con procesador detenido (25 grados), constantes por velocidad utilizada en el consumo energético (0.25 para Min, 0.5 para Med y 1 para Max), entre otras. Se propone evaluar e implementar la funcionalidad necesaria para que un usuario pueda editar estas constantes globales.
- El módulo Log del Procesador permite visualizar en forma de gráficos y tablas toda la información de log registrada durante la ejecución del algoritmo. Su objetivo es facilitar el proceso de análisis del resultado obtenido para la ejecución de cada algoritmo. Este módulo puede ser explotado extendiendo la funcionalidad actual para brindar al usuario mayor información. A modo de ejemplo se propone la incorporar el cálculo y visualización de métricas de rendimiento y energéticas.
- Por último mencionar que es posible ampliar las características de cada robot para representar aún con mayor grado de detalle los procesadores actuales. Se pueden incorporar nuevas técnicas de ajustes de rendimiento o adaptar las existentes para que trabajen con algoritmos de mayor complejidad. Este enfoque de trabajo está destinado a alumnos de materias avanzadas con un grado de conocimiento previo en conceptos de concurrencia y paralelismo.

ANEXOS

Anexo I: Encuesta alumnos Taller de Programación

Indique su respuesta marcando una X en el círculo correspondiente:

1. La aplicación ECMRE colabora en el aprendizaje de los conceptos presentados.

En completo desacuerdo	En desacuerdo	Ni de acuerdo, ni en desacuerdo	De acuerdo	Completamente de acuerdo
◯	◯	◯	◯	◯

Observaciones:

_____
_____
_____

2. Es útil contar con una herramienta de aplicación práctica que permita mediante ejemplos concretos visualizar los contenidos teóricos aprendidos en clase.

En completo desacuerdo	En desacuerdo	Ni de acuerdo, ni en desacuerdo	De acuerdo	Completamente de acuerdo
◯	◯	◯	◯	◯

Observaciones:

_____
_____
_____

3. Los contenidos en ECMRE se encuentran organizados y su utilización es intuitiva.

En completo desacuerdo	En desacuerdo	Ni de acuerdo, ni en desacuerdo	De acuerdo	Completamente de acuerdo
◯	◯	◯	◯	◯

Observaciones:

_____
_____
_____

4. La iconografía y gráficos utilizados en la aplicación tienen el tamaño adecuado y coinciden con la función asociada.

En completo desacuerdo	En desacuerdo	Ni de acuerdo, ni en desacuerdo	De acuerdo	Completamente de acuerdo
◯	◯	◯	◯	◯

Observaciones:

_____
_____
_____

5. Es beneficioso contar con esta aplicación como complemento a las clases teóricas.

En completo desacuerdo	En desacuerdo	Ni de acuerdo, ni en desacuerdo	De acuerdo	Completamente de acuerdo
◯	◯	◯	◯	◯

Observaciones:

_____

_____

_____

Referencia bibliográfica

- ACM/IEEE-CS. Joint Task Force on Computing Curricula. “Computer Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering”. Report in the Computing Curricula Series. 2004.
- ACM/IEEE-CS. Joint Interim Review Task Force. “Computer Science Curriculum 2008: An Interim Revision of CS 2001”. Report from the Interim Review Task Force. 2008.
- ACM/IEEE-CS. Joint Task Force on Computing Curricula. “Computer Science Curricula 2013”. Report from the Task Force. 2013.
- ACM/IEEE-CS. Joint Task Group on Computer Engineering Curricula. “Computer Engineering Curricula 2016”. A Report in the Computing Curricula Series. 2015.
- ACM. http://www.acm.org/. 2017.
- Akl S. The Design and Analysis of Parallel Algorithms. Prentice-Hall, Inc. 1989.
- AMD. Tecnología Turbo Core de AMD. Disponible en http://www.amd.com/es-xl/innovations/software-technologies/turbo-core. [Fecha de consulta 08/01/2017].
- Altube A., Sattolo I., Lipera L. Bot guía para un ambiente virtual inmersivo. 2016.
- Andrews G. “Foundations of Multithreaded, Parallel and Distributed Programing” Addison Wesley Higher Education 2000.
- Area, M. Los medios y las tecnologías en la educación. 2004.
- Balladini J.; Grosclaude E.; Hanzich M.; Suppi R.; Rexachs D.; Luque E. Incidencia de los modelos de programación paralela y escalado de frecuencia de CPUs en el consumo energético de los sistemas de HPC. Disponible en http://hdl.handle.net/10915/18919 [Fecha de consulta 18/12/2016]. 2010.
- Balladini J.; Uribe F.; Suppi R.; Rexachs D.; Luque E. "Factores influyentes en el consumo energético de los Sistemas de Cómputo de Altas Prestaciones basado en CPUs y GPUs". Facultad de Informática, Universidad Nacional del Comahue, Argentina, y Departamento de Arquitectura de Computadores y Sistemas Operativos, Universidad Autónoma de Barcelona, España. XVII Congreso Argentino de Ciencias de la Computación. 2011.
- Balladini J.; Casanova B.; Morán M; Uribe F.; De Giusti A. Aspectos energéticos de sistemas de computación. Disponible en http://hdl.handle.net/10915/27288 [Fecha de consulta 18/12/2016]. 2013.
- Balladini J.; Morán M.; Rozas C.; De Giusti A.; Suppi R.; D. Rexachs; Luque E. Consumo energético de sistemas de Computación de Altas Prestaciones. 2015.
- Becker Weber, B. Teaching CS1 with Karel the Robot in Java. Conference Proceddings Thirtysecond SIGCSE Technicalk Symposium on Computer Science Education. 2001.
- Belloch O. C. Las Tecnologías de la Información y Comunicación (T.I.C.). Universidad de Tecnología Educativa/Universidad de Valencia. Disponible en http://www.uv.es/~bellochc/pdf/pwtic1.pdf [Fecha de consulta 05/02/2017]. 2012.
- Beneforti M. F., Ainchil M. V. Hipermedia Aplicada a la Educación. 2000.
- Bower F. A.; Sorin D. J.; Cox L. P. The Impact of Dynamically Heterogeneous Multicore Processors on Thread Scheduling. IEEE Computer Society. 2008.

- Burks W.; Goldstine H. H., and Von Neumann J. “Preliminary discussion of the logical design of an electronic computing instrument”. 1946.
- Cabero, J. A. Impacto de las nuevas tecnologías de la información y la comunicación en las organizaciones educativas. En Lorenzo, M. y otros (coords): Enfoques en la organización y dirección de instituciones educativas formales y no formales (pp. 197-206). Granada, Grupo Editorial Universitario. 1998
- Cabero J. A. Tecnología Educativa. 2007.
- Calzadilla M. E. Aprendizaje colaborativo y tecnologías de la información y la comunicación. 2002.
- Champredonde R., De Giusti A. Herramienta visual para la enseñanza de programación. 1996.
- Champredonde R., De Giusti A. Design and Implementation of the Visual DaVinci Language. 1997.
- Champredonde R., Pasini A., Mauriello A., De Giusti A. Un Lenguaje de Programación Visual. 1999.
- Chapman B. The Multicore Programming Challenge, Advanced Parallel Processing Technologies; 7th International Symposium, (7th APPT'07), Lecture Notes in Computer Science (LNCS), Vol. 4847, p. 3, Springer-Verlag (New York). November 2007.
- Dann W., Cooper S. Education: Alice 3: concrete to abstract. Communications of ACM 52, 8, 27–29. 2009.
- De Giusti A., Madoz C., Lanzarini L. Algoritmos, datos y programas. 2001.
- De Giusti A., Madoz C., Gorga G., Feierherd G., Depetris B. Enfoques y herramientas en la enseñanza de un primer curso de computación (CS1). 2003.
- De Giusti A.; Frati F. E.; Leibovich F.; Sanchez M.; De Giusti L.; Madoz M. C. "Concurrencia y Paralelismo en CS1: la utilización de un Lenguaje Visual orientado.", VII Congreso de Tecnología en Educación y Educación en Tecnología. 8 p. http://hdl.handle.net/10915/18451. Junio 2012a.
- De Giusti A., Frati E., Leibovich F., Sanchez M., De Giusti L., Madoz M. C. LMRE: Un entorno multiprocesador para la enseñanza de conceptos de concurrencia en un curso CS1. 2012b.
- De Giusti A.; Tinetti F.; Naiouf M.; Chichizola F.; De Giusti L.; Villagarcía H.; Montezanti D.; Frati E.; Pousa A.; Rodriguez I.; Eguren S.; Denham M.; Iglesias L.; Mendez M. Arquitecturas Multiprocesador en Cómputo de Altas Prestaciones: Software de Base, Métricas y Aplicaciones. Disponible en http://hdl.handle.net/10915/42793 [Fecha de consulta 28/01/2017]. 2014.
- De Giusti A., Naiouf M., De Giusti L., Chichizola1 F., Pousa A., Sanz V., Ismael Rodriguez I., Rucci E. Enfrentando desafíos de la curricula en Informática: Concurrencia, Paralelismo, Cloud Computing, Multicores y GPUs. 2015b.
- De Giusti L. Mapping sobre Arquitecturas Heterogéneas. Tesis Doctoral. 2008.
- De Giusti L.; Frati F. E.; Leibovich F.; Sanchez M.; Madoz. M. C. "LMRE: Un entorno multiprocesador para la enseñanza de conceptos de concurrencia en un curso CS1". TE & ET; no. 7, p. 7-15. http://hdl.handle.net/10915/18283. 2012b.

- De Giusti L., Leibovich F., Sánchez M., Chichizola F., Naiouf M., De Giusti A. Desafíos y herramientas para la enseñanza temprana de Concurrencia y Paralelismo. 2013.
- De Giusti, L. C.; Leibovich, F.; Chichizola, F.; Naiouf, M.; De Giusti, A. E. Incorporando conceptos en la enseñanza de concurrencia y paralelismo utilizando el entorno CMRE. XXI Congreso Argentino de Ciencias de la Computación. 10 pp. Disponible en http://hdl.handle.net/10915/50645 [Fecha de consulta 19/11/2016]. 2015a.
- De Giusti L., Chichizola F., Rodríguez Eguren S., Sánchez M., Paniego J. M., De Giusti A. Introduciendo conceptos de Cloud Computing utilizando el entorno CMRE. 2016.
- Depetris B. O., Mallea D. A., Pendenti H., Tejero G., Feierherd G. Experiencias con Da Vinci Concurrente en la enseñanza inicial de la programación y la programación concurrente. Disponible en http://hdl.handle.net/10915/27581 [Fecha de consulta 19/01/2017]. 2013.
- Eijkhout V.; Chow E.; Van de Geijn R. Introduction to High-Performance Scientific Computing. 2015.
- Fagin B. S. Using Ada-Based Robotics to Teach Computer Science. 2000.
- Fagin B. S., Merkle L. D., Eggers T. W. Teaching Computer Science With Robotics Using Ada/Mindstorms 2.0. 2001.
- García J. C.; Señas P.; Moroni N. Cubik: Una herramienta de apoyo en la enseñanza de la Programación. Disponible en http://hdl.handle.net/10915/24233 [Fecha de consulta 11/02/2017]. 1996.
- Ge R.; Feng X.; Pyla H.; Cameron K.; Feng W. Power Measurement Tutorial for the Green500 List. Disponible en https://www.top500.org/files/green500/tutorial.pdf [Fecha de consulta 18/12/2016]. 2007.
- Gepner P.; Kowalik M.F. “Multi-Core Processors: New Way to Achieve High System Performance”. In: Proceeding of International Symposium on Parallel Computing in Electrical Engineering 2006 (PAR ELEC 2006). Pags. 9-13. 2006.
- Global Taskforce. Harmonizing Global Metrics for Data Center Energy Efficiency. Global Taskforce Reaches Agreement Regarding Data Center Productivity. 2014.
- Gómez, Daniel. Ventajas y desventajas de las TIC en la enseñanza. Revista Científica y Tecnológica UPSE, [S.l.], v. 2, n. 2, oct. 2015. ISSN 1390-7697. Disponible en http://incyt.upse.edu.ec/revistas/index.php/rctu/article/view/45.
- Gomez Naranjo J., Redondo Castro C. Las redes sociales como fuente de conocimiento en la enseñanza. Disponible en http://www.cite2011.com/Comunicaciones/TIC/150.pdf [Fecha de consulta 23/02/2017]. 2011.
- Grama A.; Gupta A.; Karypis G.; Kumar V. "Introduction to Parallel Computing". Pearson Addison Wesley, 2nd Edition. 2003.
- Green500. http://www.green500.org/. 2016.
- Gros, B. Aprendizaje Colaborativo. 2006
- Hager G.; Wellein G. Introduction to High Performance Computing for Scientists and Engineers. H. Simon, Ed. CRC P. 2011.
- Held J.; Bautista J.; Koehl S. “From a few cores to many: A tera-scale computing research overview”. Intel Corporation. 2006.

- Hennessy J. L.; Patterson D. A. Computer Architecture. A Quantitative Approach. Elsevier, Inc. 2012.
- Hernández E. J., Uribe H. M. El paradigma de la programación visual. 2013.
- Hoonlor A.; Szymanski B. K.; Zaki M. J.; Thompson J. "An Evolution of Computer Science Research". Communications of the ACM. 2013.
- Hyari A. A Comparative Study on Heterogeneous and Homogeneous Multiprocessors. 2009.
- III-LIDI. Curso de Ingreso 2017: Expresión de Problemas y Algoritmos. Facultad de Informática UNLP. Disponible en http://weblidi.info.unlp.edu.ar/catedras/ingreso/index.html. [Fecha de consulta 15/06/2017] 2017.
- Intel. Multi-core Introduction. https://software.intel.com/en-us/articles/multi-core-introduction. 2012.
- Intel. Preguntas más frecuentes sobre la Tecnología Intel® Turbo Boost. Disponible en http://www.intel.la/content/www/xl/es/support/processors/000005641.html?wapkw=00000 5641. [Fecha de consulta 07/01/2017a].
- Intel. Tecnología Intel® Turbo Boost. Disponible en http://www.intel.la/content/www/xl/es/support/processors/000005647.html [Fecha de consulta 07/01/2017b].
- Intel. Tecnología Intel® Turbo Boost 2.0. Un mejor desempeño cuando más lo necesita. Disponible en http://www.intel.la/content/www/xl/es/architecture-and-technology/turbo-boost/turbo-boost-technology.html [Fecha de consulta 07/01/2017c].
- Intel. Preguntas más frecuentes sobre la tecnología de Intel® Turbo Boost máx. 3.0. Disponible en http://www.intel.la/content/www/xl/es/support/processors/000021587.html?wapkw=00002 1587. [Fecha de consulta 08/01/2017d].
- Kozlowicz Joe. Beyond PUE: New Metrics for Data Center Energy Efficiency. 2014.
- Lang W.; Patel J. M. Towards Ecofriendly Database Management Systems. 2009.
- Maciá C. P. La virtualización de los recursos tecnológicos, impulsor del cambio en la empresa. 2006.
- McIntyre W. A. Visual Method for Generating Iconic Programming Environment. Rensselaer Polytechnic Institute. 1992.
- Madoz M. C., Gorga G., De Giusti A. Impacto de las TICs en los procesos de Articulación, Ingreso y Aprendizaje universitario. 2005.
- Markoff J.; Lohr S. Intel's huge bet turns iffy. New York Times. 2002.
- McCool M. D. "Scalable Programming Models for Massively Multicore Processors," in Proceedings of the IEEE, vol. 96, no. 5, pp. 816-831. May 2008.
- Mesa A. Método para el manejo del balanceo de carga en sistemas de cómputo distribuido de alto desempeño. Tesis de Grado. Universidad Nacional de Colombia. 2009.

- Montes de Oca E. Comparación del uso de GPGPU y cluster de multicore en problemas con alta demanda computacional. Disponible en http://hdl.handle.net/10915/44917 [Fecha de consulta 18/04/2017]. 2012.
- Montes de Oca E.; De Giusti L.; De Giusti A.; Naiouf M. Análisis de la escalabilidad y el consumo energético en soluciones paralelas sobre cluster de multicores y GPU para un problema con alta demanda computacional. Disponible en http://hdl.handle.net/10915/31731 [Fecha de consulta 18/12/2016]. 2013.
- Moore G. E. Cramming more components into integrated circuits. Revista Electronics, Volumen 38, Nº 8. 1965.
- Moreira M. A. Introducción a la Tecnología Educativa. 2009
- Naiouf, M. R.; De Giusti, A. E. Métricas del paralelismo y balance de carga en sistemas paralelos. http://hdl.handle.net/10915/21543. 2003.
- Naiouf, M. R.; De Giusti L. C.; Chichizola F., and De Giusti A. E. Dynamic Load Balancing on Non-homogeneous Clusters. http://hdl.handle.net/10915/9604. 2006.
- Naiouf M.; De Giusti A.; De Giusti L.; Chichizola F.; Pousa A.; Sanz V.; Leibovich F.; Frati E.; Rucci E; Gallo S.; Ronchetti F.; Encinas D. Procesamiento Paralelo y Distribuido. Fundamentos, Modelos y Aplicaciones. XIII Workshop de Investigadores en Ciencias de la Computación. 2011.
- Naiouf M.; Chichizola F.; De Giusti L.; Montezanti D.; Rucci E.; Frati E.; Pousa A.; Leibovich F.; Encinas D.; Villagarcía H.; Romero F.; Montes de Oca E.; Balladini J.; De Giusti A. Tendencias en Arquitecturas y Algoritmos Paralelos. XV Workshop de Investigadores en Ciencias de la Computación. p. 690-695. Disponible en http://hdl.handle.net/10915/27294 [Fecha de consulta 08/12/2016]. 2013.
- Naiouf M.; Chichizola F.; De Giusti L.; Montezanti D.; Rucci E.; Frati E.; Pousa A.; Leibovich F.; Rodríguez I.; Eguren S.; Encinas D.; Villagarcía H.; Romero F.; Montes de Oca E.; Balladini J.; De Giusti A. Arquitectura y Algoritmos Paralelos en HPC: Tendencias Actuales. Disponible en http://hdl.handle.net/10915/46198 [Fecha de consulta 18/12/2016]. 2015.
- Olguín, E. Generalidades de la Tecnología Educativa. 2012.
- Pattis, R. Karel the Robot : A Gentle Introduction to the Art of Programming. Wiley & Sons. 1981.
- Rao T. V.; Bala Krishna A. S. V. "Evolution of Multi-core Processors" , International Journal of Latest Trends in Engineering and Technology ,Vol. 3, pp. 315- 319, September 2013.
- Rucci E. Evaluación de rendimiento y eficiencia energética de sistemas heterogéneos para bioinformática. Tesis Doctoral. 2015.
- Russo C., Durán L., Calderone M., Saenz M., Sarobe M., Alonso N., Esnaola L., Pérez D., De Vito C., Piergallini R., Segura N., Picco T., Massa G., Ramón H. Generación de Contenidos Educativos Digitales. 2014.
- Salinas, J. La investigación ante los desafíos de los escenarios de aprendizaje futuros. RED, Revista de Educación a Distancia. Número 32. 2012.

- Sanz C., Madoz C., Zangara A., Albanesi B. El trabajo colaborativo y cooperativo mediado por TICs. Herramientas informáticas utilizadas en la mediación y experiencias realizadas. 2008.
- Sanz C., Madoz C., Gorga G., Zangara A., Gonzalez A., Depetris B., Ibáñez E., Moralejo L., Martorelli S., Artola V., Sanchez M. Escenarios educativos mediados por tecnología informática. Avances y resultados en las líneas de investigación actuales. 2014a.
- Sanz C., Moralejo L., Barranquero F. Materiales del Curso de Doctorado: "Diseño y Producción de Objetos de Aprendizaje". 2014b.
- Sanz C., Madoz C., Gorga G., Gonzalez A., Zangara A., Depetris B., Ibáñez E., Moralejo L., Martorelli S., Artola V., Sanchez M. Tendencias en procesos educativos mediados por Tecnologías de la Información y la Comunicación. 2015.
- Schneider Electric, "Go Green, Save Green. The Benefits of Eco-Friendly Computing". 2008.
- Semenov A. Las tecnologías de la información y la comunicación en la enseñanza: Manual para docentes o Cómo crear nuevos entornos de aprendizaje abierto por medio de las TIC. 2005.
- Sopeña J. G.; Cuesta E. H.; Vallejo M. L. Green IT: Tecnologías para la eficiencia energética en los sistemas TI. 2008.
- Sitio web del green500: http://www.green500.org/. 2016.
- Sitio web del top500: http://www.top500.org/. 2016.
- The Green Grid. The Green Grid Data Center Power Efficiency Metrics: PUE and DCiE. 2007a.
- The Green Grid. Green Grid Metrics: Describing Datacenter Power Efficiency Technical. Committee White Paper. 2007b.
- The Green Grid. http://www.thegreengrid.org/. 2017.
- Tinetti F. G.; De Giusti A. Procesamiento Paralelo, Conceptos de Arquitecturas y Algoritmos. 1998.
- Top500: http://www.top500.org/. 2016.
- Uht A. K.; Vaccaro R. J. TEAPC: Adaptive Computing and Underclocking in a Real PC. 2004.
- Von Neumann, J. "First Draft of a Report on the edvac". Moore School of Electrical Engineering, Universidad de Pensilvania. 1945.
- Willebeek-LeMair M. H.; Reeves A. P. Strategies for Dynamic Load Balancing on Highly Parallel Computers. IEEE Transactions on Parallel and Distributed Systems archive, Volume 4 Issue 9, September 1993.