CEA Comité Español de Automática
XLVI JORNADAS AUTOMÁTICA CARTAGENA 2025
UNIVERSIDADE DA CORUÑA

Jornadas de Automática

Framework para generación de datos en sistemas de tiempo real

Fontalba, Marc[a,*], Ortiz, Luis[a], Balbastre, Patricia[a], Guasque, Ana[a], Simó, José[a], Crespo, Alfons[a]

[a] *Instituto de Automática e Informática Industrial, Universitat Politècnica de València, Camino de Vera, s/n, 46022, Valencia, España.*

To cite this article: Fontalba, M., Ortiz, L., Balbastre, P., Guasque, A., Simó, J., Crespo, A. 2025. Framework for data generation in real-time systems. Jornadas de Automática, 46. https://doi.org/10.17979/ja-cea.2025.46.12184

Resumen

La presente contribución se enmarca en el contexto de los sistemas de tiempo real particionados en entornos multiprocesador. Se presenta una herramienta que permite realizar la planificación estática de tareas en un entorno desacoplado. Esta herramienta engloba desde la generación de carga sintética con parámetros temporales configurables, hasta la validación de los planes generados. El proceso incluye las fases de asignación de las tareas a particiones, el alojamiento en procesadores de las mismas y la planificación temporal de todo el sistema.

Palabras clave: Sistemas de tiempo real, Multiprocesador, Planificación estática, Asignación, Conjunto de datos

Framework for data generation in real-time systems

Abstract

The present contribution is framed in the context of partitioned real-time systems in multiprocessor environments. A tool is presented that allows to perform static task scheduling in a decoupled environment. This tool encompasses from synthetic load generation with configurable temporal parameters, to the validation of the generated plans. The process includes the phases of assigning the tasks to partitions, hosting them in processors and the temporal scheduling of the whole system.

Keywords: Real-time systems, Multiprocessor, Static scheduling, Allocation, Dataset

1. Introducción

Los sistemas de tiempo real son aquellos en los que el correcto funcionamiento no depende únicamente del resultado lógico de sus operaciones, sino también del momento en que se producen (Davis and Burns, 2011). Es decir, deben cumplir una serie de plazos temporales definidos. El incumplimiento de estos plazos puede comprometer la validez del resultado e incluso, en sistemas críticos, ocasionar graves daños materiales e incluso pérdida de vidas humanas. En sistemas críticos, además, la planificación se realiza *offline* de forma que los instantes de ejecución de las tareas están predefinidos antes de poner el sistema en ejecución (planificación estática).

Este trabajo se centra en los sistemas de tiempo real que operan sobre plataformas multiprocesador. Estas arquitecturas presentan una mayor complejidad que sus homólogas monoprocesador debido a las incertidumbres derivadas del acceso compartido a recursos hardware y a los desafíos añadidos en la planificación y asignación eficiente de tareas (Aceituno et al., 2021). Además, se considera también una arquitectura particionada, muy utilizada en sistemas altamente críticos por su capacidad de aislamiento temporal y espacial que favorece el cumplimiento de los estrictos estándares de certificación de estos sectores (Poggi et al., 2018).

Paralelamente, la inteligencia artificial (IA) está transformando de forma acelerada una amplia variedad de sectores, como la automatización o la salud. Su capacidad para detectar patrones complejos a partir de datos y tomar decisiones informadas la convierte en una herramienta versátil para la optimización de procesos. Uno de los ámbitos que podría beneficiarse significativamente de la aplicación de la IA es el de los

*Autor para correspondencia: mfonroc@ai2.upv.es

sistemas de tiempo real, donde podría emplearse, por ejemplo, para la predicción de parámetros de ejecución, la planificación temporal o la recomendación de estrategias de asignación.

No obstante, la aplicación de modelos de aprendizaje profundo en este contexto requiere la disponibilidad de grandes volúmenes de datos representativos, los cuales no siempre son accesibles debido a la naturaleza crítica de estos sistemas. Para afrontar esta limitación, en este trabajo se propone el desarrollo de una herramienta que permite la generación de conjuntos de datos sintéticos, cubriendo todo el flujo de trabajo de planificación fuera de línea: desde la generación parametrizada de tareas, hasta su asignación a particiones y núcleos, y la planificación temporal del conjunto. Los datos generados constituyen una base sólida para la aplicación de técnicas de inteligencia artificial en este tipo de sistemas, facilitando así la investigación en ambos campos. Además, la herramienta ofrece una alternativa flexible para simular entornos reales sin depender de hardware físico.

2. Flujo de Trabajo

El framework implementado automatiza el flujo de trabajo de la planificación fuera de línea en sistemas de tiempo real particionados sobre plataformas multiprocesador. Este proceso se estructura en cinco fases secuenciales, donde la salida de cada etapa sirve como entrada para la siguiente, lo que permite una simulación coherente y realista: (1) generación de tareas, (2) asignación a particiones, (3) mapeo en núcleos, (4) planificación temporal y (5) validación del plan. La Figura 1 muestra una visión general del proceso, y en los siguientes subapartados se detallan cada una de estas fases según el orden del diagrama.

Figura 1: Flujo de trabajo de planificación temporal de un sistema particionado de tiempo real multiprocesador.

2.1. Generador de tareas

En la etapa de generación de conjuntos de tareas, se ha optado por emplear un generador sintético basado en el algoritmo UUnifast propuesto en (Davis and Burns, 2009), ampliamente utilizado para simular tareas con carga de utilización repartida de forma controlada. Los conjuntos generados están compuestos por tareas que siguen una extensión del modelo temporal de Liu and Layland (1973), al que se le han incorporado parámetros como la interferencia, definida en Aceituno et al. (2021) y la criticidad, para ajustar el modelo a un escenario más realista y acorde al entorno planteado.

Las tareas τ_i están definidas con los siguientes parámetros $\tau_i = \{C_i, D_i, T_i, I_i, Cr_i, P_i, Co_i\}$. Donde C_i es el peor tiempo de ejecución (WCET), D_i el plazo, T_i el período de activación de la tarea, I_i la interferencia, Cr_i la criticidad, P_i la partición de pertenencia, Co_i el WCET cuando se produce la interferencia.

Cabe destacar la incorporación explícita de la interferencia como parámetro adicional, ya que aporta ventajas frente a otros enfoques. En primer lugar, permite desacoplar el modelo de tareas de un hardware específico; por lo tanto, se evita realizar un análisis exhaustivo del hardware para determinar el efecto de la interferencia. En segundo lugar, se obtiene un modelo más realista y menos conservador que aquellos en los que la interferencia se añade al WCET directamente, ya que estos modelos asumen que la interferencia se produce en cada activación de la tarea, lo que no se ajusta al comportamiento real de los sistemas.

En la fase de generación, la herramienta permite ajustar una serie de parámetros que determinan las características del escenario. Estos incluyen: el número de conjuntos a generar, la cantidad de tareas por conjunto, la utilización global objetivo, el número de núcleos del sistema, el hiperperíodo de las tareas, el número de tareas con interferencias y el porcentaje de incremento del WCET cuando se produce una interferencia.

2.2. Asignación a particiones

La fase de asignación a particiones organiza los conjuntos de tareas generadas previamente para garantizar el aislamiento espacial y temporal entre particiones. Para ello, se han implementado dos enfoques de optimización: uno exacto, basado en programación lineal entera mixta (MILP) utilizando Gurobi, y otro heurístico, basado en algoritmos genéticos (GA). Ambos métodos agrupan tareas según su nivel de criticidad y aseguran que la utilización total de cada partición no exceda la capacidad de un procesador, es decir, $UT \leq 1$.

2.3. Alojamiento a núcleos

En la fase de alojamiento a núcleos se determina en qué procesador se ejecutará cada tarea, asumiendo que no habrá migración de tareas en nuestro escenario. En esta etapa, se emplean las mismas técnicas de optimización utilizadas en la fase de asignación a particiones. Sin embargo, también se incorpora un enfoque adicional, considerando el problema como un caso de bin-packing. Para ello, se han implementado diversos algoritmos heurísticos, como Best Fit (BF), First Fit (FF) y Worst Fit (WF), (Coffman Jr et al., 1984), cada uno con dos variantes: uno que ordena las tareas por utilización creciente (IU) y otro por utilización decreciente (DU). Además, se ha introducido una variante de FF y WF en orden decreciente de utilización, que también agrupa las tareas según la criticidad de las mismas, propuesta en (Ortiz et al., 2024). Al igual que en la etapa de asignación a particiones, las optimizaciones basadas en MILP y algoritmos genéticos tienen como objetivo agrupar las tareas que comparten tanto el nivel de criticidad como la partición, asegurando que la capacidad de cómputo de cada procesador no sea superada, es decir, que $UT \leq 1$.

2.4. Planificación temporal

En la etapa de planificación se construye el plan temporal de ejecución de las tareas, asignando los recursos disponibles a cada una de ellas con el objetivo de maximizar el cumplimiento de sus plazos temporales. Para ello, se han implementado tres planificadores.

En primer lugar, Earliest Deadline First (EDF), (Liu and Layland, 1973), que asigna prioridades dinámicamente en función de la proximidad del plazo de cada tarea. En segundo lugar, el Planificador Combinado (PC) propuesto por Aceituno et al. en (Aceituno et al., 2022), que alterna diferentes políticas de planificación en intervalos de tiempo definidos, buscando

reducir el efecto de la interferencia y/o los cambios de contexto. Por último, se ha desarrollado un planificador orientado a la minimización de cambios de contexto, con el objetivo de reducir la sobrecarga asociada a los conmutadores de tareas.

2.5. Validación

En la fase de validación se analizan los planes generados en la etapa de planificación con el fin de obtener una serie de métricas que caracterizan su calidad y viabilidad. Entre los aspectos evaluados se encuentra la planificabilidad del plan y si se han producido errores debidos a interferencias.

Además, se recopilan métricas adicionales como la utilización total del conjunto, el número de núcleos utilizados, la cantidad de interferencias registradas, así como el peor tiempo de respuesta (WCRT), el mejor tiempo de respuesta (BCRT) y el intervalo de acción de control (CAI)(Crespo et al., 1999). También se calcula el número medio de cambios de contexto en el sistema y el promedio de cambios de contexto por partición.

3. Conjunto de datos generado

A continuación, se describen las características del conjunto de datos generado mediante el framework propuesto. Se definieron 24 escenarios experimentales combinando variaciones en el número de núcleos, tareas, utilización del sistema, porcentaje de tareas con interferencias y el impacto de estas sobre el WCET, según se detalla en la Tabla 1.

Los parámetros de los escenarios se han definido para simular entornos realistas con distintos niveles de capacidad y carga computacional. La utilización del sistema se ha ajustado al número de procesadores, considerando valores del 55 % y 75 % de carga total. Además, para analizar la robustez de los algoritmos frente a la interferencia, se ha incrementado el número de tareas interferentes proporcionalmente al número de procesadores y se ha aumentado su impacto sobre el WCET entre un 10 % y un 30 %.

Tabla 1: Definición de los escenarios experimentales

Numero de núcleos	Utilización	Numero de tareas	Tareas con interferencias	Interferencia sobre WCET (%)	Escenario
2	1.1	4	2	10	1
				20	2
				30	3
	1.5			10	4
				20	5
				30	6
4	2.1	12	3	10	7
				20	8
				30	9
	3			10	10
				20	11
				30	12
8	4.1	20	5	10	13
				20	14
				30	15
	6			10	16
				20	17
				30	18
10	5.1	28	7	10	19
				20	20
				30	21
	7.5			10	22
				20	23
				30	24

Para cada uno de estos escenarios, se generaron dos tandas de 2000 conjuntos de tareas, que fueron procesadas mediante un total de 60 combinaciones de algoritmos, resultado del cruce de 2 algoritmos de asignación a particiones, 10 algoritmos de asignación a núcleos y 3 planificadores temporales. Las distintas combinaciones consideradas se detallan en la Tabla 2.

A cada combinación se le aplicó la validación para extraer sus métricas características, lo que dio lugar a un conjunto de aproximadamente 5,76 millones de planes generados y analizados.

Tabla 2: Definición de los escenarios experimentales

Tareas a particiones	Tareas a procesadores	Planificadores	Combinación
MILP	MILP	CC	1
		EDF	2
		PC	3
	WFDU	CC	4
		EDF	5
		PC	6
	WFIU	CC	7
		EDF	8
		PC	9
	FFDU	CC	10
		EDF	11
		PC	12
	FFIU	CC	13
		EDF	14
		PC	15
	BFDU	CC	16
		EDF	17
		PC	18
	BFIU	CC	19
		EDF	20
		PC	21
	WFDUPart	CC	22
		EDF	23
		PC	24
	FFDUPart	CC	25
		EDF	26
		PC	27
	GA	CC	28
		EDF	29
		PC	30
GA	MILP	CC	31
		EDF	32
		PC	33
	WFDU	CC	34
		EDF	35
		PC	36
	WFIU	CC	37
		EDF	38
		PC	39
	FFDU	CC	40
		EDF	41
		PC	42
	FFIU	CC	13
		EDF	44
		PC	45
	BFDU	CC	46
		EDF	47
		PC	48
	BFIU	CC	49
		EDF	50
		PC	51
	WFDUPart	CC	52
		EDF	53
		PC	54
	FFDUPart	CC	25
		EDF	56
		PC	57
	GA	CC	58
		EDF	59
		PC	60

4. Resultados

Del total de planes generados, únicamente el 27,6 % corresponde a planes realizables, lo que representa un total de 1,57 millones de planes válidos sobre los 5,76 millones analizados. La distribución de la planificabilidad por escenario se muestra en la Figura 2, donde se observa que los escenarios con un menor número de núcleos presentan, en general, un mayor índice de éxito en la planificación. Asimismo, se aprecia que el aumento de las interferencias y su impacto sobre el WCET afectan negativamente a la planificabilidad global de los conjuntos.

Al analizar la robustez de las distintas combinaciones de algoritmos frente a escenarios variados, se destaca la combinación 35 (GA/WFDU/EDF) por obtener el mayor número de planes realizables, especialmente en entornos exigentes. Este buen rendimiento se atribuye al algoritmo WFDU, que distribuye las tareas de forma equilibrada y evita la sobrecarga en los núcleos.

En contraste, la estrategia BF muestra los peores resultados, debido a su tendencia a concentrar tareas en procesadores con menor carga restante, lo que genera sobrecargas locales. A partir del escenario 6, BF deja de producir planes factibles, evidenciando su limitada efectividad bajo condiciones más complejas o con mayor carga.

En términos de optimización, se evaluó la proporción de planes planificables por combinación, así como su desempeño en métricas clave: interferencia total, cambios de contexto, cambios de contexto por partición y el CAI. En ciertos escenarios, solo las combinaciones con WF generan planes factibles, lo que restringe las comparaciones inter-algoritmo y puede inducir una aparente superioridad de WF en algunas métricas al no haber alternativas viables.

Debido a la extensión del documento, las figuras siguientes presentan solo las primeras 30 combinaciones, que se diferencian del resto únicamente por el algoritmo de asignación a particiones. Dado que no se observan diferencias significativas entre ambos enfoques, los resultados no incluidos mantienen un comportamiento similar.

En cuanto a la interferencia observada en la Figura 3, los escenarios simples muestran valores bajos para todos los algoritmos, ya que la baja proporción de tareas con interferencias permite agruparlas eficientemente. En escenarios complejos, las combinaciones con WF destacan, aunque principalmente por su mayor capacidad de generar planes factibles. En este contexto, el planificador PC supera a EDF y CC, ya que su estrategia busca explícitamente minimizar la interferencia.

En relación con el CAI, el algoritmo de asignación WF presenta los mejores resultados, en parte por su mayor capacidad para generar conjuntos planificables. Asimismo, el planificador EDF destaca por minimizar eficazmente el intervalo de acción de control, superando en esta métrica a PC y CC.

Respecto a los cambios de contexto, las combinaciones con WF tienden a obtener mejores resultados, aunque ello se debe principalmente a su mayor planificabilidad, lo que reduce el número de combinaciones comparables. Dentro de cada estrategia de asignación a procesadores, el planificador CC, diseñado específicamente para reducir cambios de contexto, es el que consistentemente logra los valores más bajos, tanto a nivel global como por partición.

En particular, en los cambios de contexto por partición mostrados en la Figura 4, se observa una clara ventaja en las combinaciones que priorizan la agrupación de tareas de una misma partición en los mismos núcleos. Esta estrategia favorece la ejecución local, reduce la fragmentación del plan y, en consecuencia, disminuye la cantidad de cambios de contexto dentro de cada partición.

Adicionalmente, se han generado diagramas de cajas para representar la distribución de las principales métricas en cada combinación de algoritmos. Estas visualizaciones permiten analizar tanto la tendencia central como la variabilidad y la presencia de valores atípicos. En muchos casos, se observa una alta dispersión y frecuencia de outliers, lo que evidencia el comportamiento caótico del problema de planificación en sistemas de tiempo real particionados. Pequeñas variaciones en las características o asignaciones de las tareas pueden generar planes con métricas muy dispares o incluso inviables. Esta alta sensibilidad a las condiciones iniciales resalta la complejidad y no linealidad del problema, justificando el uso de estrategias robustas y adaptativas, como las basadas en inteligencia artificial.

Dado la extensión del documento, se han incluido únicamente las gráficas correspondientes al CAI y a los cambios de contexto.

En la distribución de la interferencia por escenario, se aprecia que en escenarios menos exigentes el valor medio de interferencia es bajo, debido a la escasa presencia de tareas interferentes. A medida que aumenta la complejidad del escenario, la interferencia también crece, alejándose de cero. Sin embargo, en los escenarios más exigentes, la baja cantidad de planes factibles genera una menor densidad de datos, afectando también a otras métricas como el CAI y los cambios de contexto.

En el caso particular del CAI como muestra la Figura 5, su distribución por escenario es más uniforme que en otras métricas. No obstante, desde un punto de vista teórico, el CAI debería aumentar con la complejidad del escenario debido al mayor impacto de las interferencias. Esta tendencia no se observa plenamente en los resultados, en parte por la escasez de planes factibles a partir del escenario 12, lo que distorsiona su representación estadística.

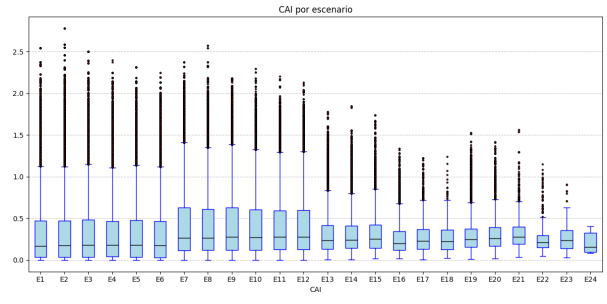

Figura 5: Distribución del valor del CAI por escenario.

Respecto a los cambios de contexto totales Figura 6, se observa una tendencia ascendente en los valores medios por escenario. Esto se explica principalmente por el aumento del número de tareas, ya que, al incrementar la carga del sistema, se incrementa también la probabilidad de que se produzcan

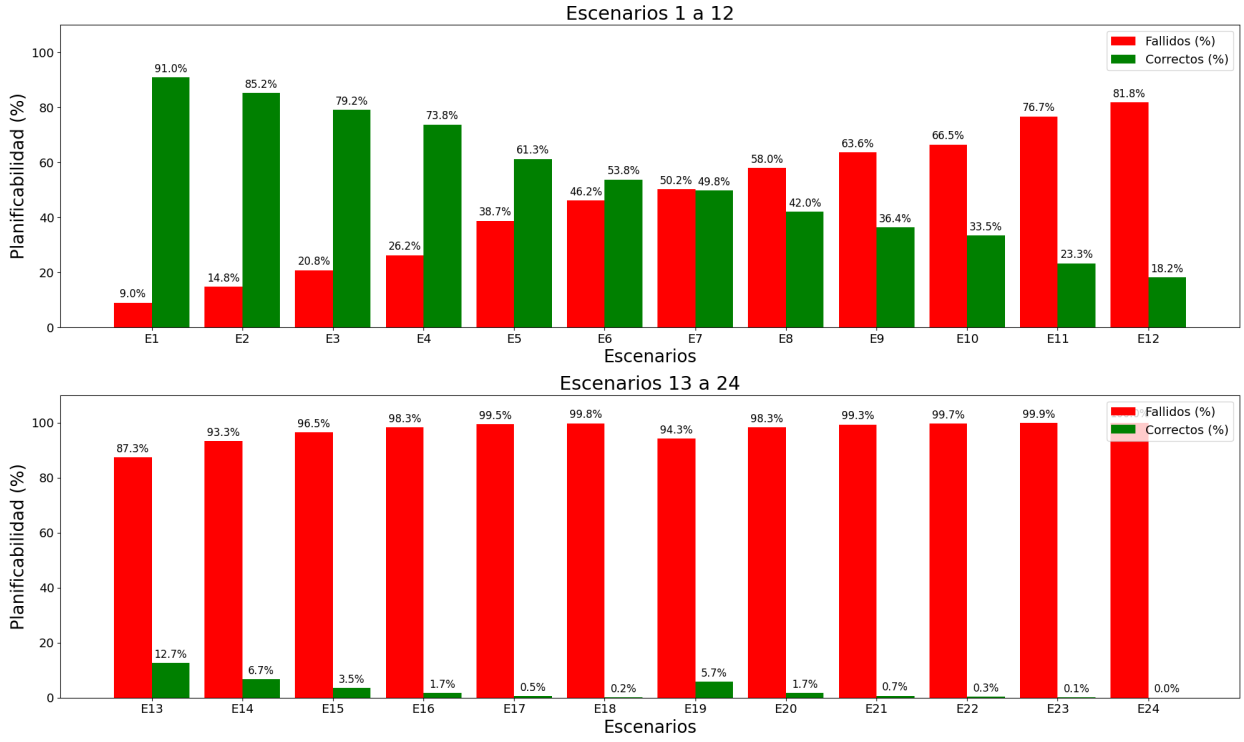

Figura 2: Distribución de la planificabilidad por escenario.

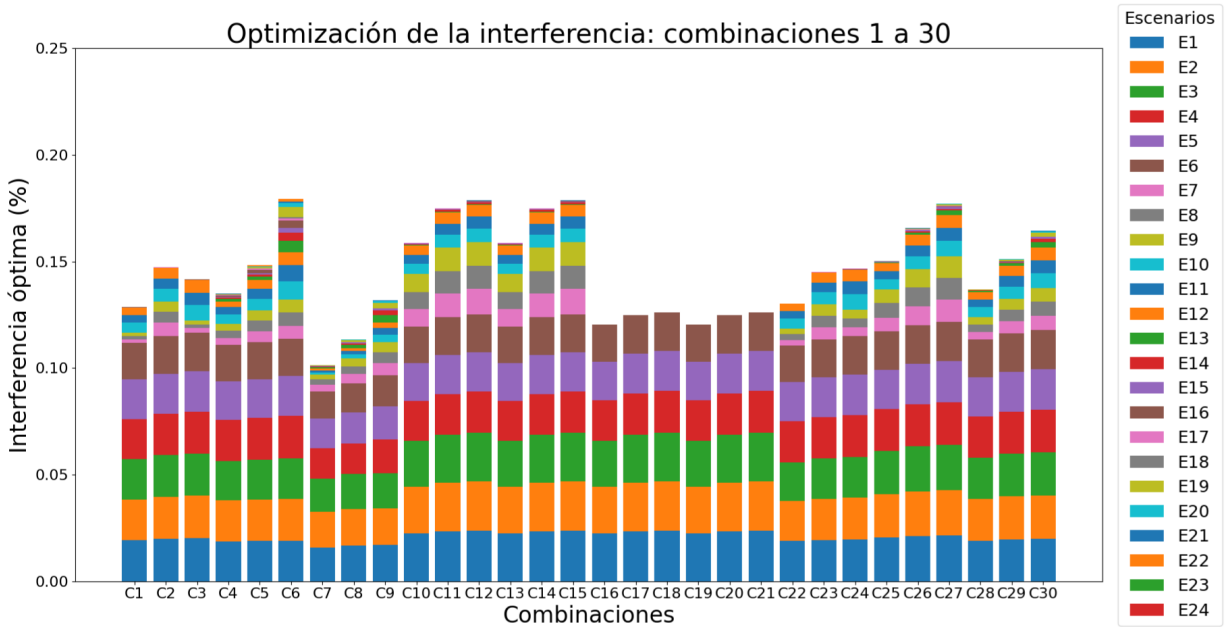

Figura 3: Distribución de interferencia óptima por algoritmo.

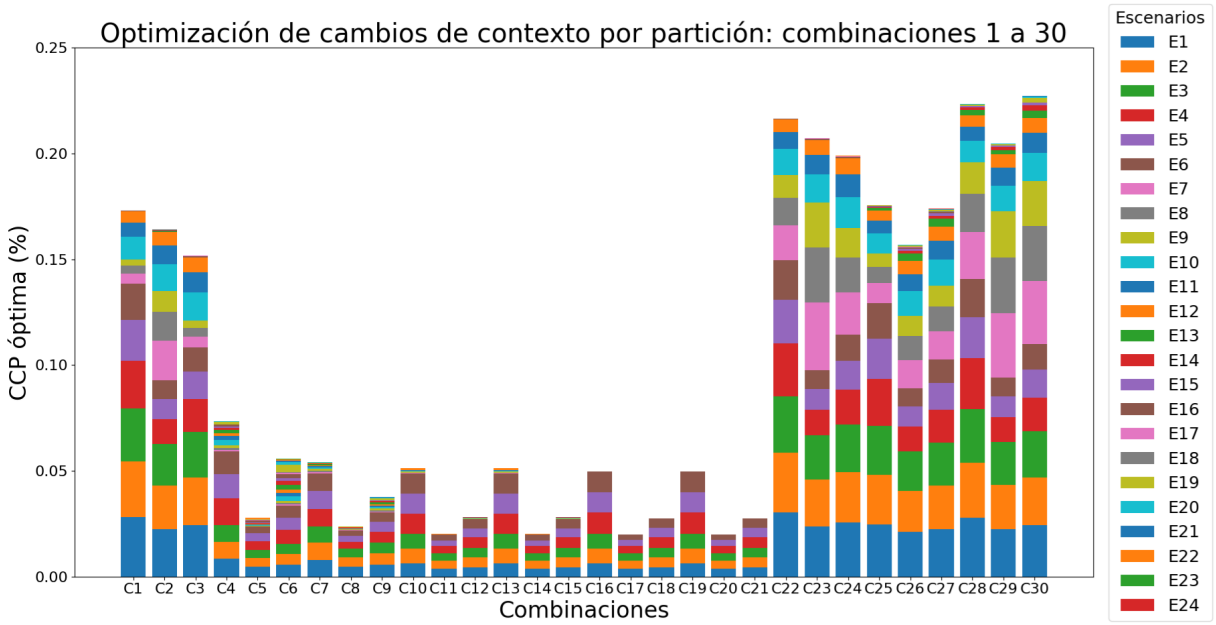

Figura 4: Distribución del cambios de contexto por partición optimo por algoritmo.

más cambios de contexto durante la planificación. Este efecto es especialmente evidente en escenarios con alta utilización, múltiples interferencias y una elevada densidad de tareas por procesador o partición.

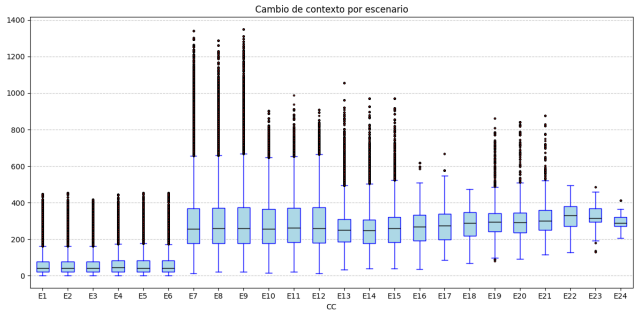

Figura 6: Distribución del valor de cambios de contexto por escenario.

5. Conclusiones

La herramienta desarrollada ha permitido la generación de un conjunto de datos que abarca los escenarios propuestos en el contexto de sistemas de tiempo real particionados sobre plataformas multiprocesador. Este framework constituye una base sólida para la construcción y extracción de datos, facilitando su posterior análisis o experimentación, tanto en el ámbito de la inteligencia artificial —por ejemplo, mediante técnicas de aprendizaje automático— como en el propio dominio de los sistemas de tiempo real, permitiendo evaluaciones comparativas, pruebas de algoritmos y estudios de comportamiento.

En adelante, esta herramienta será empleada para la generación de conjuntos de datos destinados a la creación de un entorno experimental sobre el cual se aplicarán técnicas de aprendizaje automático. El objetivo principal es el desarrollo de planificadores inteligentes, pruebas de planificabilidad y predictores de parámetros relacionados con la ejecución de algoritmos sobre diversos conjuntos de tareas.

Agradecimientos

Esta publicación es parte del proyecto de I+D+i PID2021-124502OB-C41, financiado por MCIN/ AEI/10.13039/501100011033.

Referencias

Aceituno, J., Guasque, A., Balbastre, P., Simo, J., Crespo, A., 2022. Planificador combinado: una estrategia para mejorar el rendimiento de los sistemas de tiempo real crítico. pp. 870–876.

Aceituno, J. M., Guasque, A., Balbastre, P., Simó, J., Crespo, A., 2021. Hardware resources contention-aware scheduling of hard real-time multiprocessor systems. Journal of Systems Architecture 118, 102223.

Coffman Jr, E. G., Garey, M. R., Johnson, D. S., 1984. Approximation algorithms for bin-packing—an updated survey. In: Algorithm design for computer system design. Springer, pp. 49–106.

Crespo, A., Ripoll, I., Albertos, P., 1999. Reducing delays in rt control: The control action interval. IFAC Proceedings Volumes 32 (2), 8527–8532, 14th IFAC World Congress 1999, Beijing, Chia, 5-9 July.

Davis, R. I., Burns, A., 2009. Priority assignment for global fixed priority pre-emptive scheduling in multiprocessor real-time systems. In: 2009 30th IEEE Real-Time Systems Symposium. pp. 398–409.
DOI: 10.1109/RTSS.2009.31

Davis, R. I., Burns, A., 2011. A survey of hard real-time scheduling for multiprocessor systems. ACM Comput. Surv. 43 (4).

Liu, C. L., Layland, J. W., 1973. Scheduling algorithms for multiprogramming in a hard-real-time environment. J. ACM 20 (1), 46–61.

Ortiz, L., Guasque, A., Balbastre, P., Simó, J., 2024. Allocation algorithms for multicore partitioned mixed-criticality real-time systems. PeerJ Computer Science 10.

Poggi, T., Onaindia, P., Azkarate-askatsua, M., Grüttner, K., Fakih, M., Peiró, S., Balbastre, P., 2018. A hypervisor architecture for low-power real-time embedded systems. In: 2018 21st Euromicro Conference on Digital System Design (DSD). pp. 252–259.