Universidad de Oviedo

DEPARTAMENTO DE INFORMÁTICA

Planificación inteligente de la carga de vehículos eléctricos

(Intelligent Scheduling of Electric Vehicle Charging)

Tesis Doctoral

Programa de Doctorado en Informática

Autor:
Jorge García Álvarez

Directores:
María del Camino Rodríguez Vela
Miguel Ángel González Fernández

Noviembre 2019

Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

RESUMEN DEL CONTENIDO DE TESIS DOCTORAL

1.- Título de la Tesis	
Español/Otro Idioma: **Planificación inteligente de la carga de vehículos eléctricos**	Inglés: **Intelligent Scheduling of Electric Vehicle Charging**

2.- Autor	
Nombre: **Jorge García Álvarez**	DNI/Pasaporte/NIE: **-**
Programa de Doctorado: **Informática**	
Órgano responsable: **Comisión Académica del Programa de Doctorado**	

RESUMEN (en español)

El aumento del uso de vehículos eléctricos puede tener un impacto muy positivo en la economía de los países, además de en el medio ambiente, ya que permite reducir la dependencia del petróleo. La contribución de esta tesis al desarrollo de la tecnología de vehículos eléctricos se centra en el diseño de planificadores del proceso de carga de una flota de vehículos en garajes particulares, donde cada vehículo tiene su propio punto de carga. El objetivo que nos planteamos conseguir es la minimización del retraso total de la carga de todos los vehículos con respecto a la fecha prevista para su recogida. Este problema puede ser muy complejo debido a las restricciones que imponga la infraestructura física de las estaciones de carga. Consideramos tanto la variante estática del problema, donde conocemos con antelación el instante de llegada, el tiempo de carga y el instante de recogida de los vehículos, como la variante dinámica, donde no conocemos la información de los vehículos hasta que llegan a la estación de carga. También modelamos una variante del problema que incorpora incertidumbre en los tiempos de carga y que es, por tanto, todavía más cercana a entornos reales.

La hipótesis de partida de esta tesis es que, al tratarse de un problema NP-duro, las metaheurísticas pueden ser una metodología eficaz para resolverlo. Las metaheurísticas son procedimientos de alto nivel que sirven para organizar la aplicación de métodos heurísticos específicos en la resolución de problemas complejos de optimización. Se suelen considerar como métodos que permiten muestrear el espacio de soluciones de un problema de forma inteligente, manteniendo un equilibrio entre las componentes de diversificación sobre todo el espacio de búsqueda, e intensificación sobre las regiones más prometedoras. Son métodos no exactos que están pensados para encontrar una solución aceptable en un tiempo razonable.

Diseñamos tres algoritmos de generación de planificaciones, dos para el problema determinista y uno para la variante con incertidumbre. Utilizando como base los algoritmos anteriores, desarrollamos diversas metaheurísticas, como los algoritmos genéticos, los de colonias de abejas artificiales, o GRASP, y proponemos distintas estrategias de búsqueda local. Todos los métodos propuestos contienen elementos específicamente diseñados para el problema particular.

Los algoritmos desarrollados se han evaluado sobre bancos de ejemplos existentes en la literatura, cuando ha sido posible; diseñándose bancos de ejemplos significativos en otro caso. Los resultados de la evaluación muestran el potencial de cada uno de los métodos propuestos.

Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

RESUMEN (en Inglés)

The increasing use of electric vehicles can have a highly positive impact on the global economy, in addition to the environment, since it allows to reduce dependence on oil. The contribution of this thesis to the development of electric vehicle technology focuses on the design of scheduling algorithms for the charging process of a fleet of electric vehicles in private parking lots, where each vehicle has its own charging point. The objective is the minimization of the total tardiness of all vehicles with respect to the due date indicated by the users. This problem can be very complex due to the constraints imposed by the physical infrastructure of the charging stations. We consider both the static variant of the problem, where we know in advance the arrival time, the charging time and the due date of the vehicles, and the dynamic variant, where we do not know the information of the vehicles until they arrive at the charging station. We also model a variant of the problem that incorporates uncertainty in the charging times which is, therefore, closer to real environments.

The initial hypothesis of this thesis is that, since it is a NP-hard problem, metaheuristics can be effective methodologies to solve it. Metaheuristics are high-level procedures that organize the application of specific heuristic methods in order to solve complex optimization problems. They are usually regarded as methods that sample the solution space of a problem in an intelligent way, maintaining a balance between the components of diversification over the entire search space, and intensification on the most promising regions. They are non-exact methods that are designed to find an acceptable solution in a reasonable time.

We designed three algorithms for generating schedules, two for the deterministic problem and one for the variant with uncertainty. Using these algorithms as a basis, we have developed several metaheuristics, such as genetic algorithms, artificial bee colonies, or GRASP, and propose different local search strategies. All proposed methods contain elements specifically designed for this particular problem.

The proposals have been evaluated on existing benchmarks, when possible; designing new benchmarks otherwise. The results show the potential of the proposed methods.

SR. PRESIDENTE DE LA COMISIÓN ACADÉMICA DEL PROGRAMA DE DOCTORADO EN INFORMÁTICA

Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

FORMULARIO RESUMEN DE TESIS POR COMPENDIO

1.- Datos personales solicitante	
Apellidos: **García Álvarez**	Nombre: **Jorge**

Curso de inicio de los estudios de doctorado	2016

	SI	**NO**
Acompaña acreditación por el Director de la Tesis de la aportación significativa del doctorando	X	
Acompaña memoria que incluye		
Introducción justificativa de la unidad temática y objetivos	X	
Copia completa de los trabajos *	X	
Resultados/discusión y conclusiones	X	
Informe con el factor de impacto de las publicaciones	X	
Se acompaña aceptación de todos y cada uno de los coautores a presentar el trabajo como tesis por compendio	X	
Se acompaña renuncia de todos y cada uno de los coautores a presentar el trabajo como parte de otra tesis de compendio		X

* Ha de constar el nombre y adscripción del autor y de todos los coautores así como la referencia completa de la revista o editorial en la que los trabajos hayan sido publicados o aceptados en cuyo caso se aportará justificante de la aceptación por parte de la revista o editorial

FOR-MAT-VOA-033

Artículos, Capítulos, Trabajos

Trabajo, Artículo 1

Titulo (o título abreviado)	Metaheuristics for solving a real-world electric vehicle charging scheduling problem
Fecha de publicación	31/01/2018
Fecha de aceptación	08/01/2018
Inclusión en Science Citation Index o bases relacionadas por la CNEAI (indíquese)	Science Citation Index
Factor de impacto	4.873

Coautor2 X Doctor □ No doctor . Indique nombre y apellidos	Miguel Ángel González Fernández
Coautor3 X Doctor □ No doctor . Indique nombre y apellidos	Mª Camino Rodríguez Vela
Coautor4 □ Doctor □ No doctor . Indique nombre y apellidos	
Coautor5 □ Doctor □ No doctor . Indique nombre y apellidos	
Coautor6 □ Doctor □ No doctor . Indique nombre y apellidos	
Coautor7 □ Doctor □ No doctor . Indique nombre y apellidos	

Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

Trabajo, Artículo 2

Titulo (o título abreviado)	Genetic fuzzy schedules for charging electric vehicles
Fecha de publicación	17/05/2018
Fecha de aceptación	13/05/2018
Inclusión en Science Citation Index o bases relacionadas por la CNEAI (indíquese)	Science Citation Index
Factor de impacto	3.518

Coautor2 X Doctor □ No doctor . Indique nombre y apellidos	Inés González Rodríguez
Coautor3 X Doctor □ No doctor . Indique nombre y apellidos	Mª Camino Rodríguez Vela
Coautor4 X Doctor □ No doctor . Indique nombre y apellidos	Miguel Ángel González Fernández
Coautor5 X Doctor □ No doctor . Indique nombre y apellidos	Sezin Afsar
Coautor6 □ Doctor □ No doctor . Indique nombre y apellidos	
Coautor7 □ Doctor □ No doctor . Indique nombre y apellidos	

Trabajo, Artículo 3

Titulo (o título abreviado)	Electric Vehicle Charging Scheduling by an Enhanced Artificial Bee Colony Algorithm
Fecha de publicación	14/10/2018
Fecha de aceptación	11/10/2018
Inclusión en Science Citation Index o bases relacionadas por la CNEAI (indíquese)	Science Citation Index
Factor de impacto	2.707

Coautor2 X Doctor □ No doctor . Indique nombre y apellidos	Miguel Ángel González Fernández
Coautor3 X Doctor □ No doctor . Indique nombre y apellidos	Mª Camino Rodríguez Vela
Coautor4 X Doctor □ No doctor . Indique nombre y apellidos	Ramiro Varela Arias
Coautor5 □ Doctor □ No doctor . Indique nombre y apellidos	
Coautor6 □ Doctor □ No doctor . Indique nombre y apellidos	
Coautor7 □ Doctor □ No doctor . Indique nombre y apellidos	

Trabajo, Artículo 4

Titulo (o título abreviado)	
Fecha de publicación	
Fecha de aceptación	
Inclusión en Science Citation Index o bases relacionadas por la CNEAI (indíquese)	
Factor de impacto	

Coautor2 □ Doctor □ No doctor . Indique nombre y apellidos	
Coautor3 □ Doctor □ No doctor . Indique nombre y apellidos	
Coautor4 □ Doctor □ No doctor . Indique nombre y apellidos	
Coautor5 □ Doctor □ No doctor . Indique nombre y apellidos	

Coautor6 □ Doctor □ No doctor . Indique nombre y apellidos	
Coautor7 □ Doctor □ No doctor . Indique nombre y apellidos	

Trabajo, Artículo 5

Titulo (o título abreviado)	
Fecha de publicación	
Fecha de aceptación	
Inclusión en Science Citation Index o bases relacionadas por la CNEAI (indíquese)	
Factor de impacto	

Coautor2 □ Doctor □ No doctor . Indique nombre y apellidos	
Coautor3 □ Doctor □ No doctor . Indique nombre y apellidos	
Coautor4 □ Doctor □ No doctor . Indique nombre y apellidos	
Coautor5 □ Doctor □ No doctor . Indique nombre y apellidos	
Coautor6 □ Doctor □ No doctor . Indique nombre y apellidos	
Coautor7 □ Doctor □ No doctor . Indique nombre y apellidos	

Trabajo, Artículo 6

Titulo (o título abreviado)	
Fecha de publicación	
Fecha de aceptación	
Inclusión en Science Citation Index o bases relacionadas por la CNEAI (indíquese)	
Factor de impacto	

Coautor2 □ Doctor □ No doctor . Indique nombre y apellidos	
Coautor3 □ Doctor □ No doctor . Indique nombre y apellidos	
Coautor4 □ Doctor □ No doctor . Indique nombre y apellidos	
Coautor5 □ Doctor □ No doctor . Indique nombre y apellidos	
Coautor6 □ Doctor □ No doctor . Indique nombre y apellidos	
Coautor7 □ Doctor □ No doctor . Indique nombre y apellidos	

En caso de compendio de un número de artículos superior a seis, se incorporarán hojas suplementarias conforme a este modelo

A mi familia

Agradecimientos

A mis directores de tesis, María del Camino Rodríguez Vela y Miguel Ángel González Fernández, por su paciencia, confianza y apoyo durante todos estos años. Sin duda alguna esta tesis no sería posible sin sus sabios consejos y su gran dedicación. También me gustaría dar las gracias a todos los miembros del grupo iScOp de la Universidad de Oviedo por su ayuda durante este tiempo, especialmente a Inés González Rodríguez, Ramiro Varela Arias y Jorge Puente Peinador.

Resumen

El aumento del uso de vehículos eléctricos puede tener un impacto muy positivo en la economía de los países, además de en el medio ambiente, ya que permite reducir la dependencia del petróleo. La contribución de esta tesis al desarrollo de la tecnología de vehículos eléctricos se centra en el diseño de planificadores del proceso de carga de una flota de vehículos en garajes particulares, donde cada vehículo tiene su propio punto de carga. El objetivo que nos planteamos conseguir es la minimización del retraso total de la carga de todos los vehículos con respecto a la fecha prevista para su recogida. Este problema puede ser muy complejo debido a las restricciones que imponga la infraestructura física de las estaciones de carga. Consideramos tanto la variante estática del problema, donde conocemos con antelación el instante de llegada, el tiempo de carga y el instante de recogida de los vehículos, como la variante dinámica, donde no conocemos la información de los vehículos hasta que llegan a la estación de carga. También modelamos una variante del problema que incorpora incertidumbre en los tiempos de carga y que es, por tanto, todavía más cercana a entornos reales.

La hipótesis de partida de esta tesis es que, al tratarse de un problema NP-duro, las metaheurísticas pueden ser una metodología eficaz para resolverlo. Las metaheurísticas son procedimientos de alto nivel que sirven para organizar la aplicación de métodos heurísticos específicos en la resolución de problemas complejos de optimización. Se suelen considerar como métodos que permiten muestrear el espacio de soluciones de un problema de forma inteligente, manteniendo un equilibrio entre las componentes de diversificación sobre todo el espacio de búsqueda, e intensificación sobre las regiones más prometedoras. Son métodos no exactos que están pensados para encontrar una solución aceptable en un tiempo razonable.

Diseñamos tres algoritmos de generación de planificaciones, dos para el problema determinista y uno para la variante con incertidumbre. Utilizando como base los algoritmos anteriores, desarrollamos diversas metaheurísticas, como los algoritmos genéticos, los de colonias de abejas artificiales, o GRASP, y proponemos distintas estrategias de búsqueda local. Todos los métodos propuestos contienen elementos específicamente diseñados para el problema particular.

Los algoritmos desarrollados se han evaluado sobre bancos de ejemplos existentes en la literatura, cuando ha sido posible; diseñándose bancos de ejemplos significativos en otro caso. Los resultados de la evaluación muestran el potencial de cada uno de los métodos propuestos.

Summary

The increasing use of electric vehicles can have a highly positive impact on the global economy, in addition to the environment, since it allows to reduce dependence on oil. The contribution of this thesis to the development of electric vehicle technology focuses on the design of scheduling algorithms for the charging process of a fleet of electric vehicles in private parking lots, where each vehicle has its own charging point. The objective is the minimization of the total tardiness of all vehicles with respect to the due date indicated by the users. This problem can be very complex due to the constraints imposed by the physical infrastructure of the charging stations. We consider both the static variant of the problem, where we know in advance the arrival time, the charging time and the due date of the vehicles, and the dynamic variant, where we do not know the information of the vehicles until they arrive at the charging station. We also model a variant of the problem that incorporates uncertainty in the charging times which is, therefore, closer to real environments.

The initial hypothesis of this thesis is that, since it is a NP-hard problem, metaheuristics can be effective methodologies to solve it. Metaheuristics are high-level procedures that organize the application of specific heuristic methods in order to solve complex optimization problems. They are usually regarded as methods that sample the solution space of a problem in an intelligent way, maintaining a balance between the components of diversification over the entire search space, and intensification on the most promising regions. They are non-exact methods that are designed to find an acceptable solution in a reasonable time.

We designed three algorithms for generating schedules, two for the deterministic problem and one for the variant with uncertainty. Using these algorithms as a basis, we have developed several metaheuristics, such as genetic algorithms, artificial bee colonies, or GRASP, and propose different local search strategies. All proposed methods contain elements specifically designed for this particular problem.

The proposals have been evaluated on existing benchmarks, when possible; designing new benchmarks otherwise. The results show the potential of the proposed methods.

Índice general

I Memoria **1**

1. Introducción **3**
1.1. La estación de carga 5
1.2. Revisión bibliográfica 6

2. Objetivos **9**

3. El problema de planificación de la carga de vehículos eléctricos **11**
3.1. El problema determinista 11
3.1.1. Versión estática 11
3.1.2. Versión dinámica 13
3.2. El problema con incertidumbre 14
3.2.1. Aritmética de TFNs 15
3.2.2. Valor esperado 15
3.2.3. Definición de un TFN sujeto a un conjunto de restricciones . . 16
3.2.4. Definición del problema 16

4. Resolución del problema **19**
4.1. Esquemas de generación de planificaciones 19
4.1.1. Generador de planificaciones básico 19
4.1.2. Generador de planificaciones avanzado 19
4.1.3. Generador de planificaciones para el problema con incertidumbre 21
4.2. Algoritmos de resolución 23
4.2.1. Reglas de prioridad 25
4.2.2. Algoritmo genético 26
4.2.3. Algoritmo GRASP 28
4.2.4. Algoritmo memético 29
4.2.5. Algoritmo híbrido de colonia de abejas artificiales 30
4.2.6. Algoritmo genético difuso 33

5. Estudio experimental **35**
5.1. El problema determinista 35
5.1.1. Banco de instancias 35
5.1.2. Visualizador gráfico de planificaciones 36
5.1.3. Resultados 41
5.2. El problema con incertidumbre 53
5.2.1. Banco de instancias 53
5.2.2. Resultados del fGA 54

6. Conclusiones **59**
6.1. Aportaciones de la tesis 59
6.2. Trabajo futuro 60

Bibliografía **63**

II Compendio de publicaciones **71**

7. Listado de publicaciones **73**
7.1. Publicaciones en revistas indexadas en el Journal Citations Report (JCR) 73
7.2. Publicaciones en congresos relevantes según el GII/GRIN/SCIE Conference Rating 74
7.3. Publicaciones en otros congresos 74

8. Publicaciones en revistas indexadas en el Journal Citations Report (JCR) **75**
8.1. Metaheuristics for solving a real-world electric vehicle charging problem 75
8.2. Genetic fuzzy schedules for charging electric vehicles 93
8.3. Electric vehicle charging scheduling by an enhanced artificial bee colony algorithm 107

9. Publicaciones en congresos relevantes según el GII/GRIN/SCIE Conference Rating **129**
9.1. A genetic algorithm for scheduling electric vehicle charging 129

Índice de figuras

1.1. Distribución de red de la estación de carga 5

4.1. Diagrama de flujo del algoritmo b-SGS 19
4.2. Diagrama de flujo del algoritmo a-SGS 21
4.3. Ejemplo para ilustrar el algoritmo a-SGS 22
4.4. Ejemplo para ilustrar el algoritmo f-SGS 24
4.5. Diagrama de flujo del GA 26
4.6. Diagrama de flujo del algoritmo GRASP 28
4.7. Diagrama de flujo del MA 30
4.8. Diagrama de flujo del algoritmo hABC 31

5.1. Ejemplo de planificación de una instancia de un problema 37
5.2. Ejemplos de instancias del escenario 1 con $N = 20$ y $\Delta = 0.2$ 38
5.3. Ejemplos de instancias del escenario 1 con $N = 30$ y $\Delta = 0.4$ 39
5.4. Ejemplos de instancias del escenario 1 con $N = 40$ y $\Delta = 0.6$ 40
5.5. Evolución del mejor tardiness y del tardiness medio de la población de una ejecución del GA 42
5.6. Comparación de las diferentes estrategias de búsqueda local 44
5.7. Evolución del mejor tardiness y del tardiness medio de la población de una ejecución del MA 46
5.8. Comparación de las diferentes configuraciones del MA 46
5.9. Porcentaje de mejora de GRASP y MA respecto a PD 48
5.10. Tiempo de ejecución y número de EVs en cada subproblema de una instancia dinámica 50
5.11. Evolución del mejor tardiness y del tardiness medio de la población de una ejecución del algoritmo hABC 51
5.12. Evolución promedio del fitness para el fGA difuso y el algoritmo heurístico 55
5.13. Boxplot de los resultados de 30 ejecuciones del fGA y del algoritmo heurístico en dos instancias representativas 57

Índice de tablas

3.1. Resumen de la notación de la versión estática del problema 12

5.1. Resumen de los porcentajes y distribuciones de probabilidad utilizadas para generar los tiempos de llegada 36
5.2. Resumen de los porcentajes y distribuciones de probabilidad utilizadas para generar las cargas iniciales y los instantes de recogida 36
5.3. Valores probados para elegir los parámetros del GA 41
5.4. Comparación entre los operadores de cruce propuestos para un conjunto de 24 instancias 42
5.5. Comparación del GA propuesto con el algoritmo PD del estado del arte 43
5.6. Comparación de los algoritmos de generación de planificaciones 44
5.7. Comparación del tardiness y tiempos de ejecución al variar el porcentaje de individuos evaluados 46
5.8. Comparación con el estado del arte en las instancias del escenario 1 . 47
5.9. Comparación con el estado del arte en las instancias del escenario 2 . 48
5.10. Comparación con el estado del arte en las instancias del escenario 3 . 49
5.11. Resumen de los valores probados para ajustar los parámetros del algoritmo hABC 51
5.12. Comparación de hABC con el estado del arte. Escenario 1 52
5.13. Comparación de hABC con el estado del arte. Escenario 2 53
5.14. Comparación del hABC con el estado del arte. Escenario 3 54
5.15. Mejores valores del tardiness total para el fGA y el algoritmo heurístico 56
5.16. Media de la esperanza del tardiness del fGA y del algoritmo heurístico 57

Parte I

Memoria

Capítulo 1

Introducción

En los últimos años ha aumentado el interés por la investigación y el desarrollo de tecnologías que permitan combatir el impacto ambiental y la huella de carbono provocados por la dependencia de la gasolina y los combustibles fósiles [40]. El uso de vehículos eléctricos (en inglés Electric Vehicles, EVs) puede reducir esta dependencia, ya que la electricidad se puede generar de diferentes formas, incluyendo las energías renovables [86]; es más, algunos autores señalan que los EVs se pueden utilizar para compensar las fluctuaciones en la generación de energía renovable [7, 47]. Por todo ello, los EVs, que no hace mucho tiempo parecían algo futurista son, cada día mas, parte de nuestro paisaje.

El uso de EVs también presenta retos tecnológicos, como por ejemplo decidir la localización de las estaciones de carga [89] o el desarrollo de sistemas de gestión de la carga para optimizar la distribución del consumo de electricidad mientras se satisface la demanda de carga de los EVs [41]. De hecho, el desarrollo de sistemas de carga inteligentes que eviten picos de demanda demasiado elevados se considera uno de los desafíos en la tecnología de EVs [13]. Hay muchos enfoques en la literatura que intentan aumentar el número de EVs que se cargan en las horas valle de la noche con el objetivo de reducir costes [20, 59, 85]. Sin embargo, el cálculo de una buena planificación que minimice los costes y/o maximice la satisfacción de los usuarios puede ser difícil según cuales sean las restricciones físicas particulares de cada estación de carga o, simplemente, por la cantidad limitada de energía disponible.

En esta tesis consideramos un problema de planificación de la carga de un conjunto de EVs, que fue inicialmente presentado en [37, 38], y que tiene en cuenta las características de una estación de carga modelada para un garaje de una comunidad de propietarios, diseñada por el Instituto Tecnológico de Castilla y León (ITCL). En esta estación de carga los EVs tienen que cargarse mientras están estacionados en su plaza de aparcamiento con un punto de carga que está conectado con una de las tres líneas de la fuente de alimentación trifásica. La estación tiene algunas restricciones tecnológicas, como por ejemplo, que la energía contratada es limitada y por lo tanto el número máximo de EVs que se pueden cargar simultáneamente también lo es. Además, el consumo de energía debe estar equilibrado entre las diferentes líneas de la fuente de alimentación. Esta restricción de equilibrio es la diferencia más relevante respecto a otros modelos de la literatura. Estas restricciones hacen que sea difícil planificar los tiempos de carga de forma que todos los usuarios estén satisfechos, y por lo tanto es necesario un sistema de control inteligente, como se señala en [72].

En una primera versión de la definición de los problemas se supone que el conjunto de EVs debe cargarse dentro de un horizonte de planificación conocido de antemano y, para cada vehículo, también se suponen conocidos con antelación la hora de llegada, la hora de salida deseada y el tiempo necesario para cargar la bateria al 100 %. El objetivo es proporcionar una planificación que permita cargar todos los EVs de una forma factible, en el sentido de que se cumplan todas las restricciones técnicas y que se minimice el retraso total con respecto a las horas de salida deseadas. Esta versión estática del problema es de gran importancia, ya que constituye la base para resolver una versión dinámica en la que no se conocen con antelación los datos de los EVs.

Además, se supone que los tiempos de carga son exactos, un supuesto que, junto al hecho de que sean conocidos de antemano, podría considerarse poco realista. Por lo tanto, una de las propuestas que realizamos en esta tesis, orientada a reducir la brecha entre el modelo académico y lo que podría considerarse una situación real, es la incorporación de incertidumbre en los tiempos de carga. Concretamente, modelamos los tiempos de carga inciertos utilizando conjuntos difusos.

Hay algunos ejemplos de utilización de conjuntos difusos en problemas relacionados con la carga de EVs. Sin embargo, se utilizan principalmente para modelar preferencias o restricciones blandas, y para tomar decisiones en problemas con múltiples criterios, por ejemplo, para decidir la selección óptima de la localización de estaciones de carga de EVs [31] o para coordinar la carga de EVs en una red eléctrica [32]. Más relacionado con este trabajo, en [2] se establece un modelo borroso de incertidumbre en el mercado de la electricidad con el objetivo de optimizar la oferta coordinada de EVs utilizados como servicios auxiliares en la red eléctrica. Por lo que sabemos, ésta es la única propuesta en la literatura sobre la utilización de conjuntos difusos para modelar parámetros inciertos en problemas relacionados con los EVs, por lo que el nuestro es el primer intento de planificar la carga de los EVs considerando incertidumbre en los tiempos de carga y modelando esa incertidumbre mediante conjuntos difusos.

En resumen, nos enfrentamos a un problema inspirado en un caso industrial, no solo académico, que es NP-duro [38]. Para este tipo de problemas se recomiendan las metaheurísticas, ya que son capaces de obtener buenos resultados en un tiempo de cálculo razonable. Además, las metaheurísticas son mejores cuando llevan incorporada una búsqueda local [21, 67, 82]. Los algoritmos que presentamos en esta tesis estan diseñados específicamente para el problema en cuestión y, en principio, no son adecuados para resolver otros problemas de planificación de EVs. La principal contribución del trabajo es, por tanto, la propuesta de métodos competitivos para encontrar soluciones viables y eficientes para este problema utilizando tiempos de ejecución razonables.

El resto de esta memoria está organizado de la siguiente forma. En las siguientes secciones de este capítulo describimos la estación de carga y realizamos una revisión bibliográfica. En el Capítulo 2 enumeramos los objetivos de esta tesis. En el Capítulo 3 describimos las versiones estática y dinámica del problema, así como la que considera incertidumbre en los tiempos de carga. En el Capítulo 4 presentamos los algoritmos propuestos para resolver los problemas descritos. En el Capítulo 5 incluimos la descripción de los bancos de ejemplos, y los resultados obtenidos por los algoritmos propuestos. Finalmente, en el Capítulo 6 resumimos las aportaciones de la tesis y planteamos algunas ideas para trabajos futuros.

1.1. La estación de carga

En esta sección se resumen las características principales de la estructura eléctrica y el modo de funcionamiento de la estación de carga. Está diseñada para ser instalada en un garaje privado donde cada plaza tiene un único propietario [73]. En la Figura 1.1 se muestran la estructura general y los distintos componentes de la estación, que está alimentada por una fuente de alimentación que proporciona energía eléctrica trifásica. Cada plaza tiene un punto de carga que está conectado a una de las fases, que cuando está activa transfiere corriente a una potencia constante (230 V y 2.3 kWh). El garaje dispone de 180 plazas en total, 60 conectadas a cada fase.

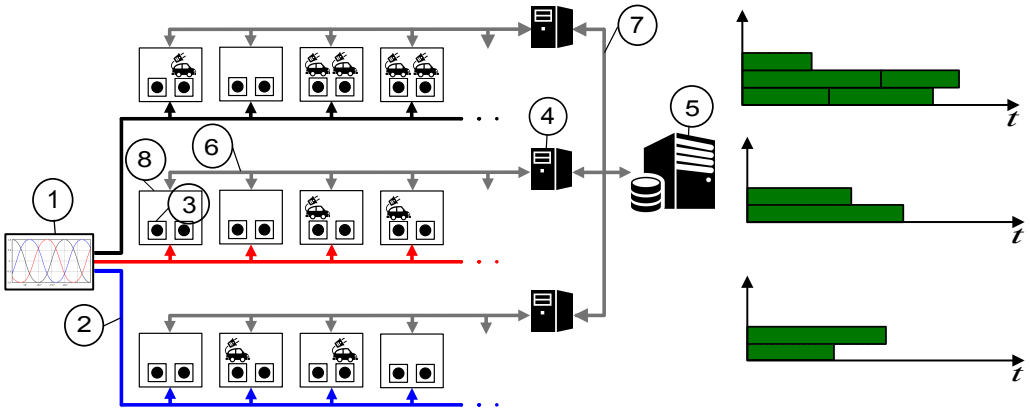


Figura 1.1: Distribución de red de la estación de carga. (1) fuente de alimentación, (2) energía eléctrica trifásica, (3) puntos de carga, (4) maestros, (5) servidor con el sistema de control y la base de datos, (6) comunicación RS 485, (7) comunicación TCP/IP, (8) esclavos. El gráfico de Gantt indica los intervalos de carga de los vehículos en cada fase.

La estación es controlada por un sistema distribuido modelo maestro-esclavo, cuyos componentes principales son:

- Un servidor con un sistema de control y una base de datos que almacena todos los datos de los vehículos y la planificación de carga.
- Un determinado número de maestros en cada fase, los cuales reunen información de los esclavos, y les envían las señales de conexión y desconexión en función de los datos recibidos por el sistema de control.
- Un esclavo por cada dos puntos de carga consecutivos en la misma fase. Los esclavos activan y desactivan los puntos de carga, y también recogen los eventos asíncronos, como por ejemplo la llegada de un nuevo vehículo al garaje.

Cuando un propietario llega a la estación tiene que registrar el tiempo de carga y la hora a la que va a recoger su vehículo. Con estos datos el sistema de control debe crear una planificación, es decir, establecer el intervalo de carga de cada vehículo, de tal forma que el retraso total, es decir, la suma del retraso de cada vehículo respecto de la hora de recogida prevista, sea mínimo.

Para crear una planificación el sistema debe tener en cuenta algunas restricciones. La potencia contratada es limitada, y por lo tanto un número máximo de vehículos

que pueden estar cargandose al mismo tiempo en una fase dada. Además, la potencia consumida por las tres fases debe estar balanceada en todo momento por razones económicas y electrotécnicas. Se sabe que un sistema desbalanceado causa grandes pérdidas de energía y tiene una eficiencia muy baja. También hay regulaciones en España (BOE, 22 de Septiembre de 2013) que no permiten sistemas muy desbalanceados a no ser que haya un consentimiento expreso de la compañía suministradora de energía, e incluso en ese caso el cliente puede ser penalizado. Otra restricción es que cada usuario es propietario de una plaza, y por lo tanto su vehículo debe estar conectado a un punto de carga concreto en una fase dada. Nótese que este hecho, combinado con la restricción de desbalanceo, hace que el problema sea realmente dificil de resolver, ya que no se pueden evitar los desbalanceos distribuyendo los vehículos entre las fases. En [73] se pueden encontrar más detalles de la estación de carga.

1.2. Revisión bibliográfica

En los últimos años se han propuesto varios modelos de planificación de carga de EVs en la literatura, lo que demuestra el interés de la comunidad investigadora. En [22, 68] podemos encontrar revisiones exhaustivas de los métodos de optimización existentes para infraestructuras de carga de EVs.

Las estrategias para la carga de EVs se pueden dividir en dos categorías [59]. En la estrategia descentralizada o distribuida, los propietarios de EVs tienen autoridad para tomar decisiones sobre el horario y la tasa de carga de sus EVs. El operador de la red puede imponer algunos incentivos de precios para mover las tareas de carga a los valles del perfil de carga, como se hace por ejemplo en [8]. Aunque esta estrategia ofrece más flexibilidad a los propietarios de EVs, puede que no garantice la optimización en la carga de EVs [59, 81] y además puede producir algunos problemas de seguridad en la red eléctrica. Por otro lado, en las estrategias de carga centralizada un sistema de control toma las decisiones de planificación de manera centralizada para tratar de obtener la solución óptima. Por lo tanto, el EV puede ser conectado por su propietario en cualquier momento, pero la hora de inicio y de finalización de la tarea de carga la decide de forma remota el sistema de control, que trata de beneficiar simultaneamente tanto a los clientes como a la empresa de distribución [45]. Para encontrar la solución óptima, el sistema de control recibe el estado en tiempo real del nivel de carga de la batería de los EVs, junto con su llegada y los tiempos de salida deseados especificados por los propietarios de los EVs [18]. La mayoría de los autores proponen una estrategia centralizada, aunque también podemos encontrar algunos ejemplos de sistemas de control descentralizados en [16, 87, 91].

Respecto a los objetivos, algunos trabajos intentan minimizar el coste total [35], mientras que otros también intentan maximizar la satisfacción de los propietarios de los vehículos además del beneficio del operador del estacionamiento [60]. Muchos autores intentan minimizar las demandas máximas y la congestión de la red llenando los valles de consumo de energía [19, 20, 52, 59, 79, 91]. Otro objetivo interesante es minimizar la diferencia entre la energía comprada en el mercado y la energía consumida por los EVs [76]. Cuando se utiliza un flujo de potencia trifásico, a menudo se considera como objetivo minimizar el desequilibrio de carga entre las fases [56]. Como podemos ver, hay muchos objetivos interesantes y, por lo tanto, algunos trabajos consideran problemas multiobjetivo. Por ejemplo, en [90] se minimizan tanto

los costes operativos totales como las emisiones. En [15] se propone un sistema de planificación de carga de EVs en tiempo real que intenta optimizar al mismo tiempo la varianza total de la carga y las preferencias de los propietarios. También, en [15] los autores coordinan la carga de EVs monofásicos con comportamiento dinámico en sistemas de distribución trifásicos desequilibrados. Se minimizan dos funciones objetivo: el coste total de la compra de energía en un entorno con precios variables y también las pérdidas totales de energía en la red durante el período de carga. Otro ejemplo se puede encontrar en [32], donde los autores minimizan los costes asociados con la generación de energía y las pérdidas de la red al tiempo que maximizan la potencia entregada a los vehículos.

Las restricciones también varían, ya que diferentes estaciones de carga y entornos conducen a diferentes conjuntos de restricciones y consideraciones. Casi todos los autores consideran las limitaciones técnicas de la red de distribución y la demanda de energía de los EVs. Algunos también consideran la variación de los precios de la electricidad [30, 58, 74, 78], o la variación de la tasa de carga de los vehículos en cada franja horaria [19, 20]. En [16] los usuarios de los EVs pueden incluso adaptar su tasa de carga según sus preferencias. Otros autores también introducen la incertidumbre y los parámetros estocásticos para optimizar la estrategia de carga [43, 57], o incluso otras consideraciones, como el precio de electricidad de carga deseado o la antigüedad de la batería [39], o la posibilidad de utilizar EVs para almacenar energía con el fin de mitigar el impacto de la incertidumbre [46]. Además, en algunas estaciones de carga, el planificador puede decidir a qué plaza se asigna cada vehículo [80], mientras que en otras estaciones de carga, como la que se considera en este trabajo, cada usuario es el propietario de una plaza en particular y, por lo tanto, debe cargar su EV en esa plaza.

En la literatura podemos encontrar diferentes técnicas metaheurísticas para abordar estos problemas, por ejemplo, particle swarm optimization [88], artificial immune systems [63], gravitational search algorithms [69], estimation of distribution algorithms [77, 78], ant-based swarm algorithms [87] o fuzzy genetic algorithms [32]. Otros tipos de algoritmos incluyen linear programming [70], moving window optimization [58], game theory based optimization [74] o distributed multi-agent methods [51].

El uso de conjuntos difusos para modelar duraciones inciertas en problemas de scheduling está ámpliamente extendido. Entre otros, en [6] se minimiza el retraso máximo en un problema de scheduling de una sóla máquina con tiempos de procesamiento difusos, y en [3] se aborda el problema de la planificación de máquinas paralelas que también usa tiempos de procesamiento difusos. Además, existen numerosos trabajos centrados en problemas de shop scheduling con duraciones difusas y, en particular, en el job shop en sus múltiples variantes [1, 55]. La lógica difusa también se ha utilizado con éxito para modelar parámetros imprecisos en problemas de la vida real, como por ejemplo en un problema de asignación de la ubicación de contenedores en una estación de transporte [92].

Cabe destacar que los métodos mencionados en esta revisión bibliográfica resuelven problemas de planificación de carga de EVs diferentes del que abordamos en esta tesis y esto es debido a que las diferentes estaciones de carga tienen conjuntos específicos de restricciones y funciones objetivo. Por lo tanto, no pueden compararse de forma natural con los métodos presentados en nuestro trabajo.

Capítulo 2

Objetivos

El principal objetivo de esta tesis es resolver un problema concreto de planificación de EVs mediante la utilización de técnicas metaheurísticas. La estación de carga descrita en la Sección 1.1 ha sido modelada para ser instalada en un garaje con unas determinadas restricciones tecnológicas, y en el que cada plaza de aparcamiento tiene un único propietario, tal y como se describe en [73]. En los artículos que hemos publicado durante el desarrollo de la tesis consideramos diferentes versiones del problema. En la versión estática se planifica un conjunto de EVs que llegan a la estación de carga en un horizonte temporal conocido, también se conocen de antemano los instantes de llegada, el tiempo de carga y el instante en el que el usuario desea abandonar la estación, mientras que en la versión dinámica no se conocen de antemano esos datos. Además, en la línea de ir incorporando características cada vez más realistas, proponemos una versión del problema en el que se incorpora incertidumbre en los tiempos de carga.

Todo lo descrito anteriormente motiva la siguiente lista de objetivos para esta tesis:

1. Revisión bibliográfica del estado del arte en scheduling con restricciones flexibles y en situaciones dinámicas, así como de técnicas metaheurísticas eficientes.

2. Análisis y diseño de algoritmos de construcción de planificaciones para el problema descrito, en sus versiones estática y dinámica, sin incertidumbre.

3. Diseño de metaheurísticas híbridas que utilicen dichos algoritmos para alcanzar soluciones eficientes en tiempos de ejecución razonables.

4. Identificación y análisis de las fuentes de incertidumbre presentes en el problema, y estudio y revisión bibliográfica de las distintas opciones para su modelado.

5. Diseño de algoritmos de construcción de planificaciones y metaheurísticas para el problema con incertidumbre.

6. Evaluar los métodos propuestos en bancos de ejemplos existentes en la literatura, cuando esto sea posible, o diseñar nuevos bancos de ejemplos significativos, en otro caso.

Capítulo 3

El problema de planificación de la carga de vehículos eléctricos

3.1. El problema determinista

En esta tesis consideramos la carga de EVs en la estación descrita en la Sección 1.1. Esta estación impone restricciones duras en la planificación de la carga, especialmente por el número máximo de vehículos que pueden estar cargándose al mismo tiempo y por la necesidad de equilibrio en el consumo de las tres líneas, debidas a razones técnicas y económicas. Además de las restricciones impuestas por la estación, se considera la restricción de no-interrupción, que impide interrumpir la carga de un EV una vez que ha comenzado, y no se plantea la posibilidad de una carga parcial de las baterías a partir de un umbral de satisfacción. Por otra parte, se asigna un intervalo de carga a todos y cada uno de los vehículos que se registran en la estación de carga, sin considerar la posibilidad de no cargar algún vehículo cuando el tiempo de inicio de su intervalo de carga asignado sea posterior a la fecha de recogida proporcionada por el propietario en su llegada al garaje. Además, asumimos que los usuarios nunca van a recoger su EV antes de la fecha de recogida indicada por ellos. Todas estas consideraciones adicionales se hacen con motivo de simplificar ligeramente este complejo problema.

El problema considerado tiene una naturaleza dinámica ya que en un entorno real no se conocen con antelación ni los instantes de llegada de los vehículos, ni los tiempos de carga, ni los instantes de recogida. Tal y como se indica en [23], la versión estática del problema no es realista, pero es interesante por numerosas razones. Por ejemplo, a partir de una solución de la versión estática del problema podemos estimar cuánto se puede mejorar una solución de la versión dinámica. Por ello, en esta tesis estudiamos las dos versiones del problema.

3.1.1. Versión estática

En la versión estática del problema conocemos con antelación los instantes de llegada y los tiempos de carga de los EVs, así como sus instantes de recogida. Por lo tanto, podemos definirlo como sigue: se consideran tres líneas de carga $L_i, 1 \leq i \leq 3$, cada una con n_i puntos de carga. Puede haber un número máximo $N > 0$ de puntos de carga activos al mismo tiempo en una línea dada. A cada línea L_i llegan M_i vehículos, $v_{i1}, \ldots, v_{iM_i}$. Para cada vehículo v_{ij} conocemos su intante de llegada

Tabla 3.1: Resumen de la notación de la versión estática del problema.

Parámetro	Descripción
L_i	Líneas de carga
n_i	Número total de puntos de carga en la línea L_i
N	Número máximo de puntos de carga activos al mismo tiempo en una línea
M_i	Número de vehículos que llegan a la línea L_i
t_{ij}	Instante de llegada del vehículo v_{ij}
p_{ij}	Tiempo de carga del vehículo v_{ij}
d_{ij}	Due date del vehículo v_{ij}
st_{ij}	Tiempo de inicio de la carga del vehículo v_{ij}
C_{ij}	Instante en el que el vehículo v_{ij} termina de cargarse
$N_i(t)$	Número de puntos de carga activos en la línea L_i en el instante t
Δ	Parámetro que controla el nivel máximo de desbalanceo
$tard_S$	Tardiness total de la planificación S

$t_{ij} \geq 0$, su tiempo de carga $p_{ij} > 0$ y el instante en el que el usuario espera recoger el vehículo de la estación de carga d_{ij}, o due date. En esta formulación suponemos que tanto el instante de llegada como el de recogida son precisos y, además, son consistentes en el sentido de que el tiempo de recogida ha de permitir completar la carga del vehículo, es decir $d_{ij} \geq t_{ij} + p_{ij}$. Idealmente, la batería del vehículo debería estar completamente cargada antes del instante de recogida, aunque puede no ser posible y por lo tanto puede haber un retraso en la carga del vehículo, también conocido como tardiness.

El objetivo del problema es obtener una planificación factible S que minimice el tardiness total $tard_S$. Una planificación factible es una asignación de tiempos de inicio para las variables de decisión st_{ij} para cada vehículo $v_{ij}, 1 \leq i \leq 3; 1 \leq j \leq M_i$, de tal forma que se satisfagan todas las restricciones del problema. Formalizaremos las restricciones como sigue:

1. Cada vehículo debe empezar a cargarse después de su instante de llegada t_{ij}.

$$\forall v_{ij}\ st_{ij} \geq t_{ij} \tag{3.1}$$

2. Una vez que un vehículo v_{ij} ha comenzado a cargarse, no puede ser desconectado. Por lo tanto, el tiempo en el que finaliza su carga, denominado C_{ij}, se calcula como sigue:

$$C_{ij} = st_{ij} + p_{ij} \tag{3.2}$$

3. Puede haber un número máximo N de puntos de carga activos en una línea en un instante dado.

$$\text{máx}\, N_i(t) \leq N, \quad t \geq 0;\ 1 \leq i \leq 3 \tag{3.3}$$

   donde $N_i(t)$ denota los puntos de carga activos en la línea L_i en el instante t.

4. Se fija un desbalanceo máximo entre dos líneas cualesquiera L_i y L_j, el cual es especificado por el parámetro Δ.

$$\text{máx}\left(\frac{|N_i(t) - N_j(t)|}{N}\right) \leq \Delta, \quad t \geq 0;\ 1 \leq i \leq 3;\ 1 \leq j \leq 3 \tag{3.4}$$

La función objetivo que se ha de minimizar es el tardiness total $tard_S$, definido de la siguiente forma:

$$tard_S = \sum_{i=1}^{3} \sum_{j=1}^{M_i} \text{máx}\,(0, C_{ij} - d_{ij}) \tag{3.5}$$

3.1.2. Versión dinámica

La versión dinámica del problema se puede modelar como una secuencia de instancias $P_1, P_2, \ldots, P_k, \ldots, P_n$ del problema estático. Cada instancia P_k está compuesta por algunos vehículos que han llegado a la estación de carga pero que todavía no se han empezado a cargar y otros que ya se están cargando. Formalmente, en una instancia P_k en un instante de tiempo T_k se considera un conjunto de vehículos $\{v_{i1}, \ldots, v_{ia_i}, \ldots, v_{im_i}\}$ en cada línea L_i de la estación de carga. Para cada vehículo v_{ij} se conoce su instante de llegada t_{ij}, su tiempo de carga p_{ij} y su due date d_{ij}. Los vehículos $\{v_{i1}, \ldots, v_{ia_i}\}$ ya han empezado a cargarse pero todavía no han terminado, es decir, para todos los vehículos $v_{ij}, 1 \leq j \leq a_i$ se mantiene que $st_{ij} < T_k$ y que $C_{ij} = st_{ij} + p_{ij} > T_k$. Por otra parte, los vehículos restantes $v_{ia_i+1}, \ldots, v_{im_i}$ se han registrado en la estación de carga pero todavía no han empezado a cargarse.

En una instancia P_k, la capacidad de la línea L_i para cargar nuevos vehículos en un instante t, denotado por $M_i^k(t)$, puede calcularse como sigue:

$$M_i^k(t) = N - \sum_{j=1}^{a_i} X_{ij}(t), \quad t \geq T_k \tag{3.6}$$

donde

$$X_{ij}(t) = \begin{cases} 1 & \text{si } t < C_{ij} \\ 0 & \text{si } t \geq C_{ij} \end{cases} \tag{3.7}$$

El objetivo es obtener una planificación factible para todos los vehículos sin planificar que han llegado a la estación hasta el instante T_k. Por lo tanto, hay que asignar un instante de inicio st_{ij} a cada uno de ellos de forma que se satisfagan las restricciones descritas para el problema estático. De nuevo, la función objetivo es la minimización del tardiness total.

Como ya se ha comentado, la resolución del problema dinámico consiste en resolver una secuencia de problemas estáticos. Idealmente, se debería construir una nueva planificación cada vez que un vehículo que requiere cargarse llega a la estación. Sin embargo, en un entorno real se podría sobrecargar el sistema de control si llega un número excesivo de vehículos en un corto período de tiempo. Por ello, es más conveniente calcular nuevas planificaciones en intervalos de tiempo de duración ΔT, que en principio estará fijada en 2 minutos.

En cada instante de tiempo T_k el sistema de control comprueba si ha llegado algún vehículo desde el último intante T_{k-1}. En caso afirmativo, se crea una nueva instancia P_k, se resuelve y la nueva planificación reemplaza a la actual. En otro caso, si no ha llegado ningún vehículo nuevo se mantiene la planificación actual durante un tiempo ΔT. Nótese que se pueden asignar diferentes tiempos de inicio st_{ij} a un mismo vehículo en diferentes instancias P_k siempre y cuando el vehículo no haya empezado a cargarse. En caso de que ya haya empezado a cargarse, su tiempo de inicio st_{ij} ya no se puede modificar.

3.2. El problema con incertidumbre

En los problemas reales es frecuente encontrar diferentes fuentes de incertidumbre y que algunos datos de entrada no se conozcan con exactitud. En nuestro caso, el tiempo que necesitará un EV para cargarse completamente no se puede conocer con exactitud de antemano. Los valores utilizados hasta ahora están estimados a partir de datos históricos, sin tener en cuenta ni la dispersión ni el sesgo que puede haber en estos datos. La opción de modelar estos tiempos utlizando distribuciones de probabilidad exige disponer de datos históricos de buena calidad y, además, el resultado es un problema de excesiva complejidad computacional. En estos casos, los intervalos difusos, o fuzzy, constituyen una alternativa muy interesante [9, 10, 84], donde cada intervalo es una distribución de posibilidad, representando valores más o menos posibles para un tiempo de carga.

Aquí proponemos modelar cada tiempo de carga incierto utilizando un *número triangular difuso* o TFN, dado por un intervalo $[a^1, a^3]$ de valores posibles (su soporte) y un valor modal único $a^2 \in [a^1, a^3]$ con $\mu_{\widehat{a}}(a^2) = 1$. Por lo tanto, un TFN $\widehat{a}$ se puede denotar como $\widehat{a} = (a^1, a^2, a^3)$ y su función de pertenencia tiene la siguiente forma:

$$\mu_{\widehat{a}}(x) = \begin{cases} \frac{x-a^1}{a^2-a^1} & : a^1 \leq x \leq a^2 \\ \frac{x-a^3}{a^2-a^3} & : a^2 < x \leq a^3 \\ 0 & : x < a^1 \text{ or } a^3 < x \end{cases} \tag{3.8}$$

Hay que destacar que cualquier número real $a \in \mathbb{R}$ puede verse como un caso particular de un TFN $\widehat{a} = (a, a, a)$ con sus tres puntos iguales a a.

En la literatura podemos encontrar diferentes propuestas sobre cómo obtener modelos difusos a partir de datos históricos o de expertos. En [71] se sugiere obtener tres intervalos de confianza del tomador de decisiones: el núcleo, que contiene los valores más típicos, un segundo que contiene valores probables, y un último para el soporte, es decir, el intervalo fuera del cual los valores son físicamente inalcanzables o se pueden omitir. Este es el enfoque seguido en [17] o [34] para obtener números difusos de 6 puntos. Como se sugiere en [9], un modelo más simple es el número triangular difuso, obtenido a partir de un intervalo (que contiene los valores posibles) y un valor típico único, lo que proporciona un buen equilibrio entre expresividad y simplicidad. En nuestro caso, un experto podría proporcionar un TFN modelando el tiempo de carga para un kWh, y el tiempo de carga de cada vehículo se obtendría al multiplicar este TFN por la capacidad (en kWh) que debe cargarse.

El hecho de que las funciones de pertenencia de los números difusos (vistos como distribuciones de posibilidad) se codifiquen como intervalos de confianza, es también la base de propuestas alternativas para derivar las funciones de pertenencia del TFN a partir de datos históricos o mediciones siguiendo un enfoque no paramétrico. De los datos históricos podemos obtener el soporte $[a^1, a^3]$ y el valor modal a^2 (entendido como el valor más frecuente) de la función de distribución de probabilidad desconocida p subyacente a esos datos.

En [17] podemos encontrar algunas consideraciones sobre cómo se pueden generar TFNs a partir de datos históricos o información de expertos.

3.2.1. Aritmética de TFNs

Para manejar los tiempos de carga como TFNs, necesitamos poder calcular la suma, resta y máximo de dos TFNs. En principio, las operaciones aritméticas de TFNs pueden obtenerse extendiendo las operaciones correspondientes en números reales utilizando el Principio de Extensión [11]. Tanto la suma como la resta de dos TFNs $\widehat{a}$ y $\widehat{b}$ dan como resultado TFNs que vienen dados por:

$$\widehat{a} + \widehat{b} = (a^1 + b^1, a^2 + b^2, a^3 + b^3) \tag{3.9}$$

$$\widehat{a} - \widehat{b} = (a^1 - b^3, a^2 - b^2, a^3 - b^1) \tag{3.10}$$

Con respecto al máximo, calcular la expresión resultante puede llegar a ser muy engorroso, e incluso intratable. Además, el conjunto de TFNs no es cerrado con esta operación. En aras de la simplicidad y la trazabilidad de los cálculos numéricos, es bastante común en la literatura aproximar el máximo, ya sea utilizando un método de ranking [54], o por interpolación [17], evaluando solo la operación en los tres puntos de definición de cada TFN, es decir:

$$\text{máx}(\widehat{a}, \widehat{b}) \approx \text{máx}_I(\widehat{a}, \widehat{b}) = (\text{máx}(a^1, b^1), \text{máx}(a^2, b^2), \text{máx}(a^3, b^3)) \tag{3.11}$$

La aproximación máx_I ha sido ampliamente utilizada en la literatura de problemas de scheduling [17, 44, 53, 83]. Se pueden encontrar argumentos adicionales para apoyar esta aproximación en [66].

A menos que se indique lo contrario, y para utilizar una notación más simple, escribiremos máx al referirnos al máximo interpolado máx_I.

Hay que remarcar que, como caso particular, para un número real $b \in \mathbb{R}$, tenemos que $\widehat{a} - b = (a^1 - b, a^2 - b, a^3 - b)$ y $\text{máx}(\widehat{a}, b) = (\text{máx}(a^1, b), \text{máx}(a^2, b), \text{máx}(a^3, b))$.

3.2.2. Valor esperado

Cuando trabajamos con números difusos, a menudo es útil obtener su valor esperado, de manera similar a lo que se hace con las distribuciones de probabilidad en entornos estocásticos. En particular, para un TFN $\widehat{a}$, su valor esperado viene dado por:

$$E[\widehat{a}] = \frac{a^1 + 2a^2 + a^3}{4} \tag{3.12}$$

Obtenemos esta expresión siguiendo diferentes enfoques, como el valor esperado de un número difuso basado en conjuntos aleatorios [36] o el centro del valor medio de $\widehat{a}$ [12] entre otros.

Hay que remarcar que, para un TFN $\widehat{a}$, siempre se cumple que el valor esperado se encuentra en su soporte, es decir, $a^1 \leq E[\widehat{a}] \leq a^3$. Alternativamente, también se puede definir el valor esperado de la siguiente manera:

$$E[\widehat{a}] = a^2 + \frac{(a^3 - a^2) - (a^2 - a^1)}{4} \tag{3.13}$$

Una ventaja adicional es que permite comparar números difusos. De hecho, no existe un orden total en el conjunto de los números difusos, por lo que se han definido

(y se siguen definiendo) numerosos métodos de ordenación total o ranking [4]. En particular, el valor esperado define un método de ranking basado en índices, que induce una ordenación total $\leq_E$ en el conjunto de TFNs [17], donde para dos TFNs $\widehat{a}, \widehat{b}$,

$$\widehat{a} \leq_E \widehat{b} \text{ si y solo si } E[\widehat{a}] \leq E[\widehat{b}] \tag{3.14}$$

Además, $\leq_E$ coincide con otros métodos de ranking de la literatura que no se basan en los valores esperados, como se destaca en [65]. Además, en [4] se presenta un estudio numérico que sugiere que, para TFNs, el ranking basado en el valor esperado es muy similar a otros siete métodos de ranking, en el sentido de que el orden que inducen en una muestra de TFNs está fuertemente correlacionado.

Con el enfoque del valor esperado, no solo estamos adoptando una postura análoga a la planificación estocástica, sino que también estamos modelando el comportamiento de un tomador de decisiones moderado, así como teniendo en cuenta muchos otros métodos de ranking de la literatura, ya sea porque coinciden totalmente con $\leq_E$ para TFNs o porque producen órdenes muy similares (para más información véanse [27, 65]).

3.2.3. Definición de un TFN sujeto a un conjunto de restricciones

Al planificar la carga de los EVs, necesitaremos determinar los tiempos de inicio difusos para la carga de cada EV sujetos a un conjunto de restricciones. El siguiente resultado nos proporciona una forma de hacerlo.

Teorema 1. *Sea $e \in \mathbb{R}$ un número y $\widehat{c} = (c^1, c^2, c^3)$ un TFN tal que $E[\widehat{c}] \leq e$. Un TFN $\widehat{s} = (s^1, s^2, s^3)$ que satisface:*

$$E[\widehat{s}] = e \tag{R_0}$$
$$c^i \leq s^i, i = 1, 2, 3 \tag{$R_{1,2,3}$}$$

viene dado por la siguiente expresión:

$$\begin{array}{lll}
\textit{si } E[\widehat{c}] = e, & \widehat{s} = \widehat{c}; & \\
\textit{si } E[\widehat{c}] < e, c^3 - e \leq e - c^1, & \widehat{s} = (s^1, s^2, s^3) \textit{ donde } \begin{cases} s^3 &= \text{máx}(e, c^3), \\ s^2 &= \text{máx}(e, c^2), \\ s^1 &= 4e - 2s^2 - s^3; \end{cases} & (3.15) \\
\textit{si } E[\widehat{c}] < e, c^3 - e > e - c^1, & \widehat{s} = (s^1, s^2, s^3) \textit{ donde } \begin{cases} s^1 &= c^1, \\ s^3 &= c^3, \\ s^2 &= \frac{1}{2}(4e - s^1 - s^3). \end{cases}
\end{array}$$

En [27] podemos encontrar la demostración del teorema anterior.

3.2.4. Definición del problema

En [27] proponemos una versión del problema que incorpora incertidumbre en los tiempos de carga. Siguiendo la descripción general del problema,

formulamos el problema de planificación de carga de EVs con tiempos de carga difusos de la siguiente forma.

En una instancia de éste problema, hay tres líneas de carga $L_i, 1 \leq i \leq 3$. Cada línea L_i tiene n_i puntos de carga y hay un número máximo $N > 0$ de puntos de carga que pueden estar activos simultáneamente. La línea L_i recibe M_i EVs $v_{i1}, \ldots, v_{iM_i}$ entre el tiempo 0 y un horizonte de planificación H. Cada vehículo v_{ij} tiene una hora de llegada $t_{ij} \geq 0$, un tiempo de carga incierto $\widehat{p_{ij}} > 0$ (representado como un TFN) y una hora de recogida d_{ij}, o *due date*, en la cual, idealmente, el vehículo debe estar completamente cargado. Aquí asumimos que los instantes de llegada y recogida son tiempos precisos obtenidos del usuario. Además, asumimos que son consistentes con los tiempos de carga en el sentido de que $d_{ij} \geq t_{ij} + p_{ij}^3$. Es decir, que el instante de recogida debe permitir que el vehículo esté completamente cargado siempre que comience a cargarse tan pronto como llegue, incluso si requiere el mayor tiempo de carga posible.

Una planificación factible es una asignación de tiempos de inicio $\widehat{st_{ij}}$ para todos los EVs $v_{ij}, 1 \leq i \leq 3, 1 \leq j \leq M_i$ tal que se cumplan las siguientes restricciones:

1. Los vehículos no pueden comenzar a cargarse antes de su hora de llegada:

$$\forall v_{ij}, \quad st_{ij}^1 \geq t_{ij} \tag{3.16}$$

2. Una vez que un vehículo empieza a cargarse, no se puede desconectar antes de que termine de cargarse. Es decir, si $\widehat{c_{ij}}$ denota el momento en que v_{ij} termina de cargarse, debe cumplirse que:

$$\forall v_{ij}, \quad \widehat{c_{ij}} = \widehat{st_{ij}} + \widehat{p_{ij}} \tag{3.17}$$

3. El número de puntos de carga que están activos en una línea en cualquier instante no puede exceder un umbral determinado N. Para cualquier vehículo v_{ij}, su punto de carga está activo en el intervalo de tiempo entre $\widehat{st_{ij}}$ y $\widehat{c_{ik}}$. Por lo tanto, definimos el conjunto de EVs que pueden estar cargándose en la misma línea cuando v_{ij} comienza a cargarse como:

$$A_{ij} = \{v_{ik} : \exists h \in \{1,2,3\}\ st_{ik}^h \leq st_{ij}^h, \text{máx}(\widehat{c_{ik}}, \widehat{st_{ij}}) \neq \widehat{st_{ij}}\}, \tag{3.18}$$

y $|A_{ij}|$ denota su cardinal, entonces la restricción podemos expresarla de la siguiente forma:

$$\forall v_{ij}, \quad |A_{ij}| \leq N. \tag{3.19}$$

4. El número de puntos de carga activos debe distribuirse uniformemente entre las líneas. En el problema determinista $N_i(t)$ denota el número de puntos de carga activos en la línea L_i y $\Delta \in [0,1]$ controla el desequilibrio máximo entre dos líneas diferentes L_i y L_j, de modo que:

$$\forall t \geq 0, \ \max_{1 \leq i,j \leq 3} \left(\frac{|N_i(t) - N_j(t)|}{N} \right) \leq \Delta. \tag{3.20}$$

En una configuración determinista, consideramos que un vehículo v_{ij} está cargándose en el instante t si $st_{ij} \leq t \leq c_{ij}$. Extendemos este concepto a la configuración difusa tomando t como el TFN $\widehat{t} = (t, t, t)$, por lo que el número de puntos de carga activos en el instante $t \geq 0$ en la línea L_i, $1 \leq i \leq 3$, se define como:

$$N_i(t) = \sum_{j=1}^{M_i} \delta_{ij}(t) \tag{3.21}$$

donde

$$\delta_{ij}(t) = \begin{cases} 1 & : \text{ si } \widehat{st_{ij}} \leq_E \widehat{t} \leq_E \widehat{c_{ij}} \\ 0 & : \text{ en otro caso.} \end{cases} \tag{3.22}$$

Por lo tanto, esta restricción se refiere ahora al equilibrio de carga esperado (ya que $N_i(t)$ se define en términos de valores esperados).

El objetivo es encontrar una solución factible que minimice el tardiness total con respecto a los instantes de recogida, definido como:

$$\widehat{T} = \sum_{i=1}^{3} \sum_{j=1}^{M_i} \text{máx}(0, \widehat{c_{ij}} - d_{ij}) \tag{3.23}$$

Dado que la función objetivo $\widehat{T}$ es un TFN, las soluciones se compararán utilizando $\leq_E$.

Capítulo 4

Resolución del problema

4.1. Esquemas de generación de planificaciones

4.1.1. Generador de planificaciones básico

En esta sección explicamos el esquema de generación de planificaciones básico (en inglés Basic Schedule Generation Schema, b-SGS) utilizado en algunas de las metaheurísticas de esta tesis para el problema determinista. Este algoritmo construye una planificación S a partir de una permutación de vehículos V [23]. La Figura 4.1 muestra el diagrama de flujo de dicho algoritmo, que planifica secuencialmente todos los vehículos de la permutación V escogiendo para cada vehículo el instante de inicio más temprano posible de tal forma que se cumplan todas las restricciones descritas en la Sección 3.1, teniendo en cuenta todos los vehículos que ya se han planificado. Merece la pena mencionar que siempre es posible planificar un vehículo manteniendo todas las restricciones, ya que en el peor caso se podrá asignar como instante de inicio el tiempo de finalización del último vehículo en terminar su carga. Por lo tanto, siempre es posible generar soluciones factibles.

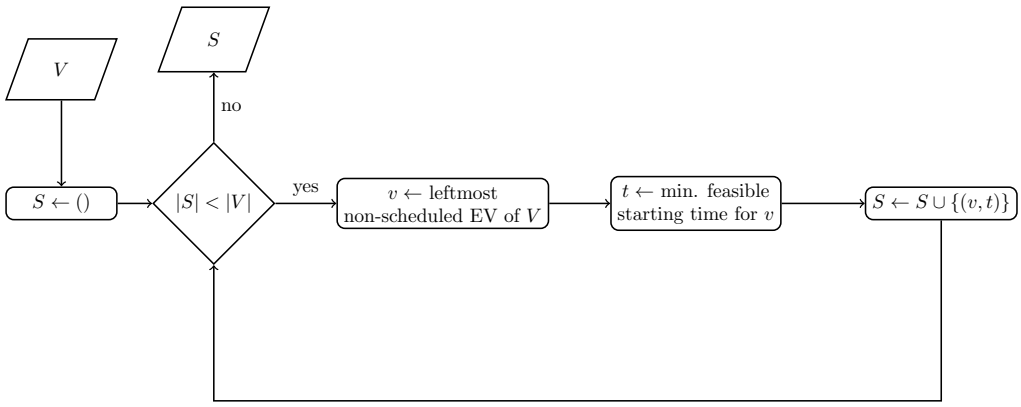


Figura 4.1: Diagrama de flujo del algoritmo b-SGS.

4.1.2. Generador de planificaciones avanzado

Como indicamos en [23], cuando utilizamos una permutación para representar las soluciones y planificamos los EVs secuencialmente en el orden indicado por la

permutación, la restricción de desbalanceo entre las líneas puede llevar a asignaciones de tiempos de inicio tardíos que hagan que un vehículo tenga un tardiness muy elevado, y por tanto hagan que esa planificación no sea buena. Para evitar esta situación hemos diseñado un esquema de generación de planificaciones avanzado (en inglés Advanced Schedule Generation Schema, a-SGS), con un procedimiento de refinamiento local que permite modificar el orden de planificación de algunos vehículos en la permutación V. Cuando planificamos la carga de un vehículo comprobamos si rompe algún bloqueo por un desbalanceo anterior, definido como sigue.

Definición 1. *Una solución parcial factible tiene un bloqueo por desbalanceo en un intervalo $[t_1, t_2)$ si y solo si existen dos líneas L y L' tales que no se puedan planificar más vehículos en L' en el intervalo $[t_1, t_2)$ por la restricción de máximo desbalanceo (ver Ecuación 3.4) con respecto a la línea L. Formalmente, $\forall t' \in [t_1, t_2)$ se mantiene que:*

$$N_{L'}(t') < N \tag{4.1}$$

$$\frac{N_{L'}(t') + 1 - N_L(t')}{N} > \Delta \tag{4.2}$$

Definición 2. *Una asignación factible de un tiempo de inicio t para un EV v en una línea L con un tiempo de carga p, rompe un bloqueo por desbalanceo en el intervalo $[t_1, t_2)$, donde $t \leq t_1 < t_2 \leq t+p$, si y solo si antes de la asignación existe un bloqueo por desbalanceo en el intervalo $[t_1, t_2)$ entre las líneas L, L', es decir, se cumplen las Condiciones 4.1 y 4.2, pero después de la asignación un nuevo vehículo se podría planificar en la línea L' en ese intervalo. Formalmente, $\forall t' \in [t_1, t_2)$ después de la asignación se cumple que:*

$$\frac{N_{L'}(t') + 1 - (N_L(t') + 1)}{N} \leq \Delta \tag{4.3}$$

$$\frac{N_{L'}(t') + 1 - N_{L''}(t')}{N} \leq \Delta \tag{4.4}$$

donde $L \neq L' \neq L''$.

Si se cumple la situación anterior en la que se rompe un bloqueo por un desbalanceo anterior, desplanificamos todos los vehículos de las líneas L' y L'' cuyo tiempo de inicio sea mayor o igual que t_2. Por lo tanto, podemos replanificarlos después en un instante anterior. Finalmente, cuando terminamos de replanificar todos los vehículos, reconstruimos la permutación inicial para que refleje el orden final en el que están planificados los vehículos. El diagrama de flujo de la Figura 4.2 muestra los detalles del procedimiento, en donde "Apply local refinement" representa el proceso de desplanificación de vehículos que hemos descrito.

Con el siguiente ejemplo ilustramos su funcionamiento: consideremos una instancia con tres vehículos (v_{11}, v_{12}, v_{13}) en la línea L_1, tres vehículos (v_{21}, v_{22}, v_{23}) en la línea L_2 y un vehículo (v_{31}) en la línea L_3. El instante de llegada es 0 para todos los vehículos excepto para v_{31}, que es $t_{31} = 5$. El tiempo de carga es 10 para todos los vehículos. El instante de recogida es 10 para los vehículos v_{11}, v_{12}, v_{21}, v_{22} y 15 para v_{13}, v_{23}, v_{31}. El número máximo N de puntos de carga activos en cualquier línea es 3 y el parámetro de desequilibrio máximo Δ es $\frac{2}{3}$ (por lo tanto, en cualquier

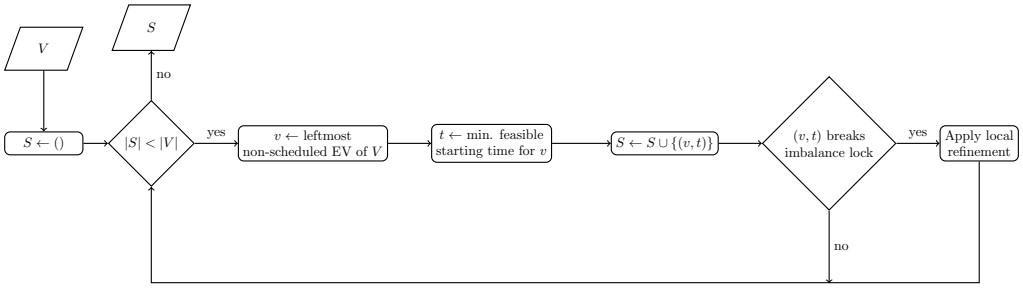


Figura 4.2: Diagrama de flujo del algoritmo a-SGS.

momento dado debe haber, como máximo, una diferencia de dos vehículos entre dos líneas cualesquiera).

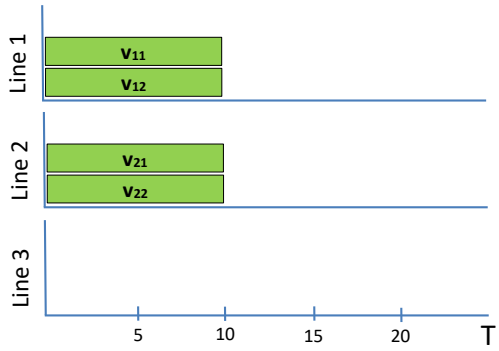
Consideremos también la permutación $(v_{11}, v_{12}, v_{21}, v_{22}, v_{13}, v_{23}, v_{31})$ y la planificación parcial donde ya se han planificado los primeros cuatro vehículos (veáse la Figura 4.3(a)). En esta situación, cuando los vehículos v_{13} y v_{23} se planifiquen, ambos empezarán en el instante 10 porque si los planificásemos en el instante 5, se violaría la restricción de desequilibrio máximo con respecto a la línea L_3. Posteriormente, podemos planificar v_{31} en el instante 5 y el resultado es la planificación de la Figura 4.3(b) con un tardiness de 10 (marcamos en rojo la parte del tiempo de carga que supera el due date). Sin embargo, podemos observar que la planificación de v_{31} rompe el bloqueo por desbalanceo en las líneas L_1 y L_2, y ahora podemos reasignar tanto v_{13} como v_{23} para comenzar a cargarse en el instante 5, como se indica en la Figura 4.3(c), para así obtener una solución sin tardiness.

4.1.3. Generador de planificaciones para el problema con incertidumbre

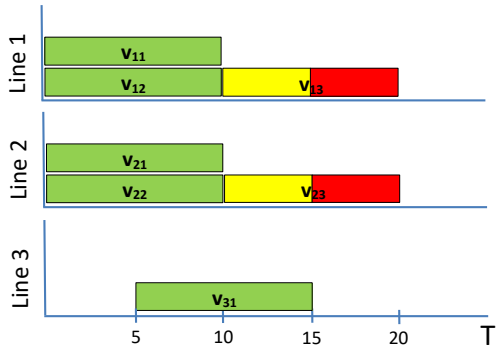
En el problema con incertidumbre debemos asignar tiempos de inicio difusos $\widehat{st_{ij}}$ a todos los EVs en el orden marcado por la permutación V. Con este fin, proponemos un esquema de generación de planificaciones difusas (en inglés Fuzzy Schedule Generation Schema, f-SGS) que tenga en cuenta las restricciones y características del problema difuso. La idea, al igual que en la versión determinista del problema, es asignar secuencialmente a cada EV el tiempo de inicio más temprano posible de tal forma que las restricciones del problema se mantengan con respecto a la planificación parcial ya existente. En el Algoritmo 1 se pueden ver los pasos detallados.

Este algoritmo divide cada línea L_i en N sublíneas virtuales l_i^k, $1 \leq k \leq N$, donde N es el número máximo de puntos de carga activos permitidos en cada línea. Haciendo esto, aseguramos el cumplimiento de la restricción respecto al número máximo de puntos de carga activos.

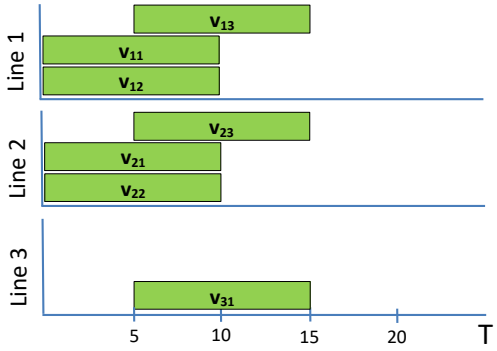
A continuación, asignamos los tiempos de inicio de carga a los vehículos en el orden en el que aparecen en la permutación V. Para cada vehículo v_{ij} calculamos el tiempo de inicio más temprano posible en cada sublínea l_i^k, que denotamos $\widehat{est}^k$, como el máximo entre su instante de llegada t_{ij} y el instante $\widehat{c}_i^k$, que es el momento en el que termina de cargarse el último vehículo ya planificado en la sublínea k. Esto garantiza que se cumpla la restricción por la que un vehículo no puede empezar a cargarse antes de su instante de llegada. Una vez hecho esto, seleccionamos la



(a) $(v_{11}, v_{12}, v_{21}, v_{22})$ están planificados.



(b) v_{13} y v_{23} se planifican en el instante 10 para cumplir la restricción de desequilibrio máximo respecto a L_3.



(c) Al planificar v_{31} se rompe el bloqueo por desbalanceo en L_1 y L_2 y se replanifican v_{13} y v_{23} en el instante 5.

Figura 4.3: Ejemplo para ilustrar el algoritmo a-SGS.

sublínea con el menor tiempo de inicio posible (de acuerdo con $\leq_E$) como candidata para planificar v_{ij}.

Antes de que se pueda asignar un tiempo de inicio a v_{ij}, debemos asegurarnos de que se mantenga el balanceo entre las líneas con respecto a los vehículos ya planificados. Esto podría implicar retrasar el inicio de la carga de v_{ij} de forma que su tiempo de inicio esperado $E[\widehat{est}^k]$ coincida con un instante st_{ij}^E posterior al valor esperado de su tiempo de inicio más temprano posible. En el caso de que el vehículo se retrase, el generador de planificaciones intenta encontrar otra sublínea donde mantener el equilibrio entre las líneas minimizando el tiempo que la sublínea está libre respecto al último vehículo planificado. Así, seleccionamos la sublínea k^* donde debemos cargar v_{ij}.

Establecemos el tiempo de inicio $\widehat{st_{ij}}$ como el tiempo difuso que podemos obtener a partir del tiempo de inicio más temprano posible $\widehat{est}^{k^*}$ en esa sublínea y el primer instante donde la carga contiene st_{ij}^E, de acuerdo con el Teorema 1 (ver Sección 3.2.3). Hay que destacar que, de acuerdo con ese resultado, $E[\widehat{st_{ij}}] = st_{ij}^E$, por lo que se mantiene el balanceo, y $\widehat{st_{ij}} = \text{máx}(\widehat{st_{ij}}, \widehat{est}^{k^*})$, por lo que el tiempo de carga es

Input Una permutación V de EVs
Output Una planificación con los tiempos de inicio de todos los EVs
Considerar cada línea L_i dividida en N sublíneas $l_i^k : k = 1 \dots N$
Inicializar el tiempo en el que el último EV de cada sublínea termina de cargarse $\widehat{c_i}^k \leftarrow 0$
for all $v_{ij} \in V$ **do**
//Seleccionar la sublínea donde v_{ij} puede empezar a cargarse antes
$\widehat{est}^k \leftarrow \text{máx}(\widehat{c_i}^k, t_{ij})$, $k = 1, \dots, N$
$k^* \leftarrow \arg\text{mín}\{E[\widehat{est}^k]\}$
//Encontrar el intervalo más temprano posible sin desequilibrio entre las líneas
$st_{ij}^E \leftarrow \text{mín}\{t \geq E[\widehat{est}^{k^*}]$: la condición de equilibrio dada en la ecuación 3.20 se mantiene en el intervalo $[t, t + \widehat{p_{ij}})\}$
//Mover a la sublínea donde, manteniendo el equilibrio, genera una menor desocupación
$k^* \leftarrow \arg\text{máx}\{E[\widehat{est}^k] : E[\widehat{est}^k] \leq st_{ij}^E\}$
//Asignar el tiempo de inicio de carga y actualizar el tiempo de fin de carga de la sublínea
$\widehat{st_{ij}} \leftarrow \text{fuzz}(\widehat{est}^{k^*}, st_{ij}^E)$
$\widehat{c_i}^{k^*} \leftarrow \widehat{st_{ij}} + \widehat{p_{ij}}$
end for
return La planificación generada

Algoritmo 1: Esquema de generación de planificaciones difusas (f-SGS).

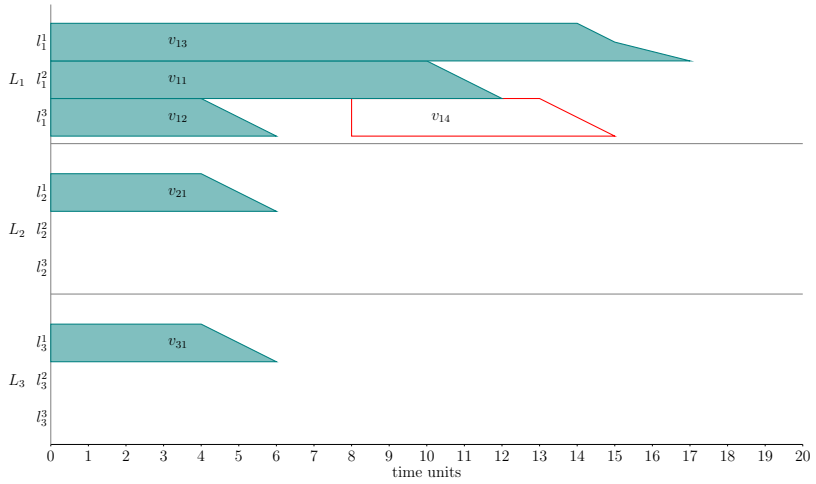
posterior a la llegada del vehículo y la restricción del número máximo de puntos de carga activos también se cumple.

Finalmente, ilustraremos el procedimiento de planificación de un vehículo mediante un ejemplo en el que debemos planificar un total de seis vehículos: cuatro vehículos $\{v_{11}, v_{12}, v_{13}, v_{14}\}$ en la línea L_1, uno $\{v_{21}\}$ en L_2 y uno $\{v_{31}\}$ en L_3. El tiempo de llegada es 0 para todos los vehículos excepto v_{14}, con $t_{14} = 8$. Los tiempos de carga vienen dados por $\widehat{p_{12}} = \widehat{p_{21}} = \widehat{p_{31}} = (4, 5, 6)$, $\widehat{p_{11}} = (10, 11, 12)$, $\widehat{p_{13}} = (14, 15, 17)$ y $\widehat{p_{14}} = (5, 6, 7)$, y los instantes de recogida son $d_{12} = d_{21} = d_{31} = 9$, $d_{11} = 11$ y $d_{13} = d_{14} = 20$. El número máximo de puntos de carga activos en cada línea es $N = 3$ y el umbral de desequilibrio máximo es $\Delta = \frac{2}{3}$, por lo que en cada instante debe haber como máximo una diferencia de 2 EVs cargándose entre dos líneas cualesquiera.

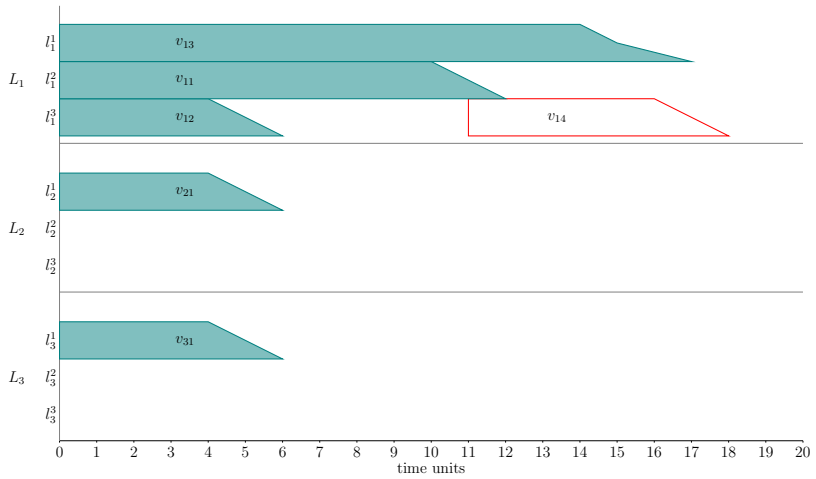
La Figura 4.4 ilustra el procedimiento de planificación si la entrada del algoritmo es la permutación $(v_{11}, v_{21}, v_{31}, v_{12}, v_{13}, v_{14})$ y todos los EVs se han planificado excepto el último (v_{14}). Para planificarlo, buscamos la sublínea k en L_1 donde v_{14} podría comenzar a cargarse lo antes posible, la que resulta ser l_1^3, con $\widehat{est}^3 = (8, 8, 8)$ (Figura 4.4(a)). Luego, calculamos el primer instante $st_{ij}^E \geq 8$ después del cual se mantiene el equilibrio entre las líneas durante todo el tiempo de carga de v_{14}, produciendo $st_{ij}^E = 11$ (Figura 4.4(b)). Luego, observamos que v_{14} se puede planificar desde ese instante en una sublínea diferente l_1^2 (Figura 4.4(c)). Por lo tanto, el valor de $\widehat{st_{14}}$ se obtiene de $st_{14}^E = 11$ de tal manera que el EV puede comenzar a cargarse después del tiempo de inicio más temprano en esa sublínea $\widehat{est^2}$, es decir, $\widehat{st_{14}} = (10, 11, 12)$.

4.2. Algoritmos de resolución

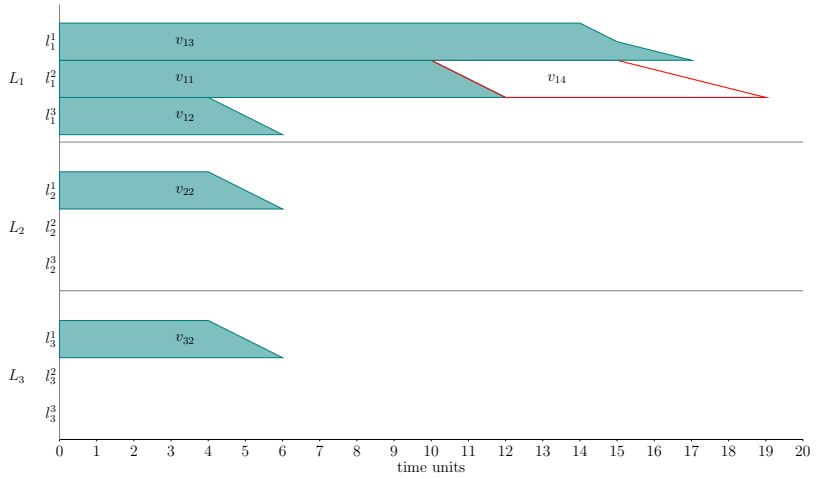
Las técnicas metaheurísticas son algoritmos aproximados de optimización y búsqueda de propósito general que permiten incorporar y explotar información específica



(a) Llegada de v_{14} a la línea L_1 en $t_{14} = 8$. El EV podría planificarse en la tercera sublínea l_1^3 con tiempo de inicio $\widehat{8}$ si no existiese la restricción de desbalanceo.



(b) El balanceo solo se mantiene a partir de $st_{14}^E = 11$.



(c) v_{14} finalmente se planifica en l_1^2 con $\widehat{st_{14}} = (10, 11, 12)$.

Figura 4.4: Ejemplo para ilustrar el algoritmo f-SGS.

del problema para tratar con objetivos complejos [14]. Proponemos su utilización debido a su capacidad para obtener soluciones de buena calidad en tiempos de ejecución razonables. Para empezar, en la Sección 4.2.1 describimos varias reglas de prioridad que incorporaremos a las metaheurísticas detalladas en las siguientes secciones.

4.2.1. Reglas de prioridad

En las metaheurísticas basadas en poblaciones necesitaremos generar un conjunto de soluciones factibles para formar una población inicial, y en las basadas en una única solución necesitaremos una nueva en cada reinicio de la misma. Estas soluciones iniciales se generan utilizando reglas de prioridad, que permiten elegir el siguiente vehículo a planificar durante la construcción de la permutación de vehículos V. Las reglas de prioridad son fáciles de implementar y consumen pocos recursos computacionales y, aunque las soluciones que proporcionan no son de altísima calidad, son un buen punto de partida para metaheurísticas complejas. Por ello, se utilizan con frecuencia en problemas de scheduling para la generación de soluciones iniciales.

En concreto, utilizamos las siguientes reglas de prioridad:

- Apparently Tardiness Rule (ATR). Se ha implementado una adaptación de la regla definida en [38]. Esta regla se ha utilizado en diferentes trabajos (ver por ejemplo [48, 75]) y es bien conocida por su efectividad para reducir el tardiness. Se propone adaptar la regla como sigue: sea $\Gamma(\alpha)$ el menor instante de tiempo de inicio de carga posible de todos los vehículos sin planificar en la planificación parcial α construida hasta el momento. Se calcula una probabilidad de selección para todos los vehículos no planificados que son capaces de empezar su carga en $\Gamma(\alpha)$ del siguiente modo:

$$\Pi_{ij} = \frac{1}{p_{ij}} \exp\left[\frac{-\max(0, d_{ij} - \Gamma(\alpha) - p_{ij})}{g\bar{p_i}}\right] \tag{4.5}$$

  donde $\bar{p_i}$ es el tiempo medio de carga de los vehículos y g es un parámetro de ajuste, que se ha fijado en $g = 0.25$, como se sugiere en [38]. Los vehículos se ordenan en orden descendente según su probabilidad Π_{ij}.

- Due Date Rule (DDR). Propuesta en [23], simplemente ordena los vehículos en orden ascendente de su due date d_{ij}.

- Latest Starting Time (LST). Propuesta en [26], clasifica a todos los vehículos en orden ascendente de sus tiempos de inicio más tardíos posibles, definidos como $lst_{ij} = d_{ij} - p_{ij}$.

- Earliest Starting Time (EST). Propuesta en [27], clasifica a todos los vehículos en orden ascendente de sus tiempos de llegada t_{ij}.

Estas reglas son capaces de construir soluciones eficientes, pero no podemos utilizar sus versiones deterministas debido a que en las metaheurísticas basadas en poblaciones es necesario construir un conjunto de soluciones iniciales diferentes, mientras que en los algoritmos basados en una única solución necesitamos utilizar

reinicios y, para cada uno, también es necesario tener un punto de partida diferente. Por lo tanto, proponemos la utilización de una versión estocástica: para seleccionar el siguiente vehículo que añadiremos a la permutación, ordenamos todos los vehículos de acuerdo a la regla de prioridad correspondiente y ejecutamos una selección por torneo, es decir, seleccionamos $tSize$ vehículos aleatoriamente y añadimos el mejor de ellos de acuerdo con la regla correspondiente. Hay que remarcar que $tSize$ es un parámetro relevante, ya que si elegimos valores pequeños se van a generar soluciones casi aleatorias, mientras que si utilizamos valores muy grandes se van a producir soluciones demasiado similares entre sí.

4.2.2. Algoritmo genético

En [23] proponemos un Algoritmo Genético (en inglés Genetic Algorithm, GA) para resolver el problema de planificación de EVs. Nuestro GA comienza generando una población inicial mediante una combinación de cromosomas aleatorios y heurísticos, que representan soluciones potenciales al problema. A continuación el algoritmo itera un cierto número de generaciones hasta que alcanza la condición de parada. En cada iteración, construye una nueva población a partir de la anterior aplicando los operadores de selección, cruce, mutación y reemplazamiento. El criterio de parada se satisface cuando el mejor cromosoma encontrado hasta el momento no mejora durante un número consecutivo de generaciones, o cuando se encuentra una solución con tardiness cero. En la Figura 4.5 mostramos el diagrama de flujo del GA, mientras que en los siguientes subapartados describimos las diferentes partes del algoritmo.

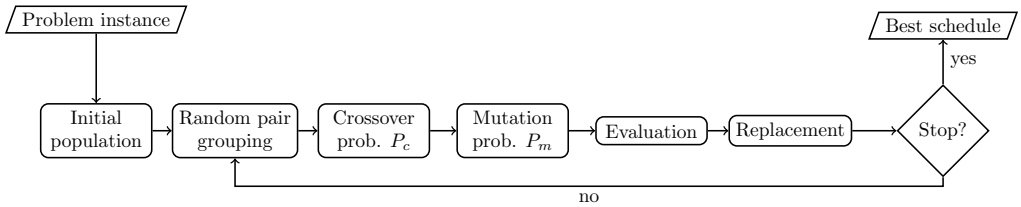


Figura 4.5: Diagrama de flujo del GA.

Codificación y población inicial

Tras analizar varias opciones, como por ejemplo utilizar una permutación para cada línea, hemos decidido codificar los cromosomas de la población como una única permutación de vehículos, lo cual nos permitió diseñar algoritmos de generación de planificaciones más simples. Como decodificador hemos utilizado el algoritmo a-SGS, descrito en la Sección 4.1.2.

Para aumentar la diversidad en la población inicial, hemos combinado la generación de cromosomas mediante dos reglas de prioridad con algunos cromosomas aleatorios, a razón de un tercio de la población de cada. Las reglas de prioridad que hemos utilizado son la ATR y la DDR, descritas en la Sección 4.2.1.

Estrategias de selección y reemplazamiento

En la fase de selección agrupamos aleatoriamente todos los cromosomas en pares. A continuación aplicamos el operador de cruce sobre cada par para generar dos descendientes, que pueden ser modificados por el operador de mutación. La estrategia de reemplazo se utiliza para elegir los cromosomas que van a pasar a la siguiente generación. Para cada par de padres y sus dos descendientes, elegimos los dos mejores cromosomas con diferente tardiness. En [24] los autores prueban que esta estrategia preserva la diversidad de la población y es capaz de mejorar los resultados del algoritmo. También hay que remarcar que la mejor solución en una población siempre es igual o mejor que la mejor solución de la población anterior.

Operadores de cruce

El operador de cruce debe generar cromosomas que hereden las mejores características de sus padres y tengan una probabilidad razonable de mejorar el valor de la función objetivo. Proponemos tres operadores diferentes:

Denotamos el primer operador como LEX (Line Exchange Crossover). Comenzamos seleccionando aleatoriamente una línea L_i, a continuación asignamos al primer descendiente todos los vehículos de la línea L_i en el mismo orden en el que aparecen en el segundo padre, y completamos con los vehículos de las dos líneas restantes en el mismo orden que aparecen en el primer padre. Creamos el segundo descendiente de forma análoga, pero intercambiando el rol de cada padre.

En el segundo operador, el SBX (Starting-time Based Crossover), seleccionamos un instante aleatorio de tiempo t_0, y generamos un primer descendiente con los vehículos del primer padre que empiezan su carga antes de t_0, y completamos el descendiente con los vehículos restantes en el orden en el que aparecen en el segundo padre. Generamos el segundo descendiente de forma análoga intercambiando los roles de los padres.

El tercer operador es estándar en la literatura. Se trata del operador de cruce en un punto (denotado 1PX) que elige aleatoriamente una posición en el cromosoma, y crea el primer descendiente eligiendo todos los vehículos de primer padre antes de esa posición, y completa el descendiente añadiendo los vehículos restantes en el mismo orden relativo que aparecen en el segundo padre. Para crear el segundo descendiente, los padres invierten sus roles.

Los operadores LEX y SBX están específicamente diseñados para resolver el problema de carga de EVs y por lo tanto deberían ser más eficientes que el operador estándar 1PX. En [23] se puede consultar la descripción detallada y algunos ejemplos de los operadores descritos.

Operador de mutación

El propósito del operador de mutación es mejorar la diversidad de la población introduciendo nuevo material genético mediante modificaciones aleatorias en algunos cromosomas. En el GA utilizamos un operador en el que cada línea se muta con cierta probabilidad. Si alguna línea L_i se muta, el operador selecciona un subconjunto aleatorio de vehículos consecutivos en esa línea, y baraja aleatoriamente su orden.

4.2.3. Algoritmo GRASP

En [24] proponemos un algoritmo GRASP (del inglés Greedy Randomized Adaptive Search Procedure). Los algoritmos GRASP son metaheurísticas comúnmente aplicadas a problemas de optimización combinatoria. Este algoritmo comienza creando una solución factible en la fase de construcción, seguida de una fase de mejora que ejecuta una búsqueda local que explora sistemáticamente la vecindad del individuo hasta que encuentra un óptimo local. Como esta metaheurística toma algunas decisiones aleatorias, repetimos el proceso $NRestarts$ veces y la salida final del algoritmo es la mejor solución de todas las repeticiones. En la Figura 4.6 mostramos el diagrama de flujo del algoritmo GRASP que describimos en detalle a continuación.

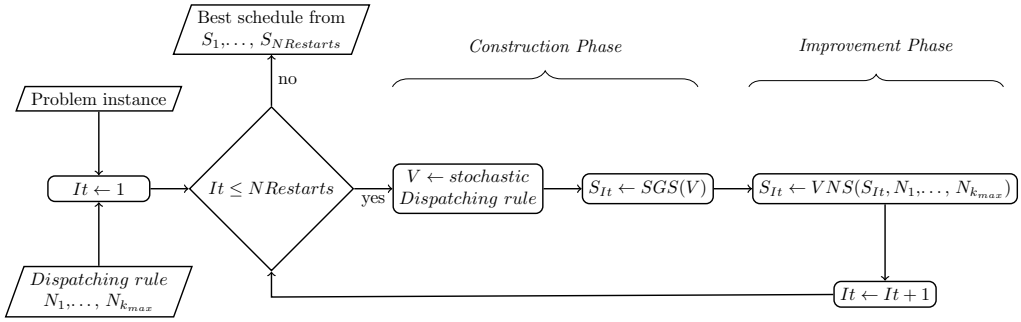


Figura 4.6: Diagrama de flujo del algoritmo GRASP.

Fase de construcción

En la fase de construcción generamos una solución inicial factible, para ello, análogamente a como se generó la población inicial en el GA, construimos una permutación V utilizando alguna de las reglas descritas en la Sección 4.2.1 en su variante estocástica, y aplicamos un algoritmo de generación de planificaciones para construir la correspondiente planificación.

Fase de mejora

En la fase de mejora implementamos una búsqueda local que empieza con una solución factible generada en la fase de contrucción e intenta mejorar su tardiness total.

La búsqueda local utiliza una escalada simple como estrategia de selección y reemplazamiento. Por lo tanto, tan pronto como evaluamos un vecino que mejora la solución actual, aceptamos el movimiento, el vecino reemplaza la solución actual, y el proceso empieza de nuevo. En otro caso descartamos el movimiento. La escalada simple continúa hasta que no aceptemos ningún movimiento durante un ciclo completo, es decir, hasta que encontremos un óptimo local.

Como estructuras de vecindad consideramos diferentes variantes de las vecindades clásicas de inserción e intercambio, y analizamos sus combinaciones para explotar un método Variable Neighborhood Search (VNS). Esta metaheurística, propuesta por Mladenovic y Hansen en [61], se basa en el cambio sistemático de vecindad hasta conseguir una solución que sea óptimo local en todas las vecindades consideradas.

En [33, 62] se realiza una revisión sobre la aplicación de VNS para resolver problemas combinatorios, y se exponen tanto los conceptos básicos de la metaheurística como los últimos avances.

La vecindad de intercambio (*Swap-Beg*) consiste, como su propio nombre indica, en intercambiar pares de elementos. Para generar una vecindad a partir de una solución, iteramos sobre la lista de vehículos empezando por el primer vehículo de la lista. Consideramos movimientos de los vehículos que se necesitan planificar antes, intentando intercambiarlos con vehículos de la misma línea que se pueden retrasar. Concretamente, para cada vehículo con tardiness positivo, consideramos intercambiarlo con otro vehículo de la misma línea sin tardiness tal que su tiempo de inicio sea menor.

La vecindad de inserción (*Ins-Beg*) es similar, pero en lugar de intercambiar los vehículos en la permutación, insertamos el vehículo con tardiness justo a la izquierda del vehículo sin tardiness.

Como utilizamos una estrategia de escalada simple, el orden en el que generamos los vecinos es claramente relevante. Por esta razón, también consideramos las vecindades de intercambio e inserción (*Swap-End* e *Ins-End* respectivamente) que iteran sobre la permutación de vehículos empezando por el último vehículo de la lista, por lo tanto, trabajamos con 4 vecindades diferentes.

Después de la generación de cada vecino, ejecutamos el algoritmo a-SGS para planificar el nuevo vecino y calcular su tardiness. Aunque hay que destacar que, como una parte de la planificación no cambia respecto a la solución original, no replanificamos los vecinos desde cero, sino que para ahorrar coste computacional, el algoritmo a-SGS empieza desde el primer vehículo de la lista que cambia debido al movimiento realizado.

Para combinar las vecindades definidas, proponemos la utilización de Variable Neighborhood Descent (VND), la cual es probablemente la variante con menor coste computacional de la VNS. Para implementarla, debemos definir una lista de vecindades $N_1, N_2, \ldots, N_{k_{max}}$. La idea es empezar ejecutando una escalada simple con la primera vecindad. Cuando alcanzamos un óptimo local con la vecindad actual, cambiamos a la siguiente vecindad, y cada vez que mejoramos la solución actual volvemos a la primera vecindad. El algoritmo termina cuando alcanzamos una solución que sea óptimo local con respecto a todas las vecindades consideradas.

4.2.4. Algoritmo memético

En [24] hemos ampliado el GA presentado en la Sección 4.2.2 para crear un Algoritmo Memético (en inglés Memetic Algorithm, MA). Para ello hemos añadido un paso adicional después de que se cumpla la condición de parada, donde aplicamos una búsqueda local a un subgrupo de cromosomas de la última generación. En la Figura 4.7 representamos el diagrama de flujo del MA.

La generación de la población inicial, y los operadores de selección, mutación y reemplazamiento son idénticos a los descritos en la Sección 4.2.2 para el algoritmo genético.

Como operadores de cruce hemos considerado dos opciones diferentes. El primer operador es el SBX, propuesto en [23] y que hemos descrito en la Sección 4.2.2. El segundo operador, el Cycle Crossover (CX), fue inicialmente propuesto en [64] y aparece con profusión en la literatura por su buen rendimiento en algoritmos gené-

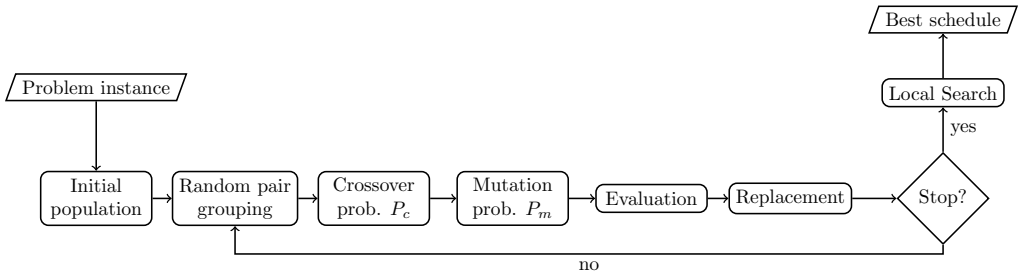


Figura 4.7: Diagrama de flujo del MA.

ticos que utilizan representaciones basadas en permutaciones. Hemos seleccionado los operadores SBX y CX ya que en un estudio preliminar hemos confirmado que proporcionan mejores resultados que LEX, que el conocido Partially-Mapped Crossover (PMX) propuesto en [29] y que el operador de cruce en un punto 1PX. Más aún, proponemos combinar los dos operadores de modo que el primer descendiente se cree con el operador SBX y el segundo con el CX. Denotamos este operador de cruce híbrido como SBX-CX. Hemos probado empíricamente que esta combinación produce mejores resultados que cada operador por separado, posiblemente porque incrementa la diversidad de la población.

En el caso del MA utilizamos el generador de planificaciones básico b-SGS en lugar del avanzado a-SGS. Hemos observado en unos experimentos preliminares que en este caso obtiene resultados similares a a-SGS pero en un menor tiempo de ejecución. Creemos que esto es debido a que las metaheurísticas avanzadas basadas en poblaciones y combinadas con búsquedas locales son capaces de encontrar buenas permutaciones por sí mismas, y por ello no es tan necesaria la utilización de un generador de planificaciones avanzado.

Búsqueda local

Es habitual combinar un GA con una búsqueda local para proporcionar una intensificación adicional en la búsqueda, lo cual suele mejorar los resultados. Nuestra principal preocupación en este punto es que evaluar una solución es computacionalmente costoso, y por lo tanto aplicar el procedimiento de búsqueda local a cada cromosoma generado, o incluso a unos pocos cromosomas de cada generación, es prohibitivo. Hemos probado varias estrategias, en las que se aplica la búsqueda local a diferentes porcentajes de cromosomas únicamente de la última generación. Concretamente se ha aplicado al 0 %, 5 % y 100 % de cromosomas y la configuración más eficiente resulta ser aplicar la búsqueda local al 5 % de cromosomas, como se verá en el estudio experimental realizado en la Sección 5.1.3.

4.2.5. Algoritmo híbrido de colonia de abejas artificiales

En [26] proponemos un algoritmo híbrido de Colonia de Abejas Artificiales (en inglés hybrid Artificial Bee Colony, hABC). Este tipo de algoritmos fueron introducidos en [49] y están inspirados en el comportamiento de las abejas, imitando la búsqueda de comida de tres tipos de abejas: trabajadora (employed), observadora (onlooker) y exploradora (scout). Estos algoritmos se suelen utilizar en problemas de scheduling debido a su efectividad y su buen equilibrio entre diversifi-

cación e intensificación. En [50] podemos encontrar una revisión de sus fundamentos y algunas aplicaciones.

En la Figura 4.8 mostramos el diagrama de flujo con las diferentes fases del algoritmo hABC propuesto, que empieza creando un número SN de soluciones iniciales o fuentes de alimento. Después, itera un cierto número de ciclos, en los que se ejecutan diferentes pasos: employed bee phase, onlooker bee phase y scout bee phase. El algoritmo satisface el criterio de parada cuando la mejor solución no mejora durante un cierto número de ciclos consecutivos, o si encuentramos una solución con tardiness cero. Finalmente, aplicamos una búsqueda local a la mejor solución encontrada.

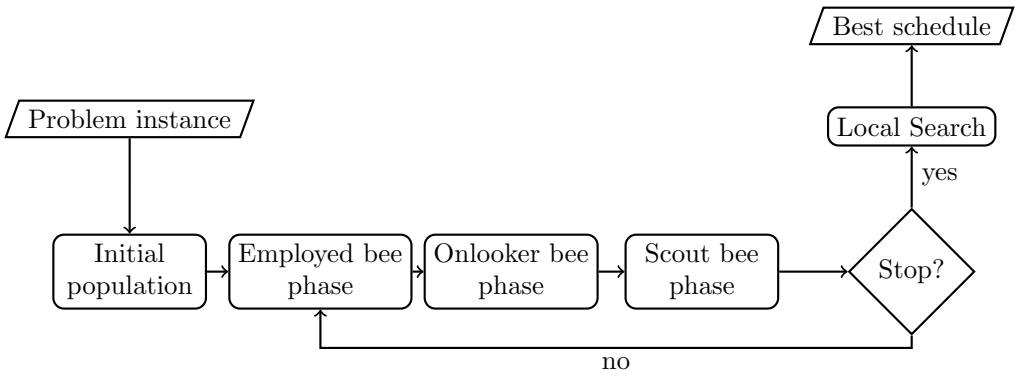


Figura 4.8: Diagrama de flujo del algoritmo hABC.

Codificamos las soluciones, o fuentes de alimento, como permutaciones de EVs de forma análoga a como lo hacíamos en los algoritmos anteriores. Cada solución tiene un valor asociado num_trials, que es el número de veces que hemos intentado mejorar esa fuente de alimento sin éxito.

Para evaluar una fuente de alimento, creamos una planificación S a partir de la permutación V con el algoritmo b-SGS. Utilizamos el generador de planificaciones básico en lugar del avanzado por las mismas razones que en el MA.

Al igual que en el GA o en el MA, proponemos combinar dos reglas de prioridad con algunas fuentes de alimento aleatorias para crear una población inicial.

Employed Bee Phase

Las abejas trabajadoras están a cargo de la búsqueda de nuevas y mejores fuentes de alimento. Para este fin, en el algoritmo ABC original, cada abeja trabajadora genera una nueva solución candidata en la vecindad de una fuente de alimento y el nuevo candidato, si fuese mejor, reemplaza la solución anterior. En el algoritmo que presentamos en esta tesis, proponemos explotar los operadores de cruce descritos en secciones anteriores siguiendo dos enfoques diferentes.

En el primero (denominado e_meth1), seleccionamos la mejor solución encontrada hasta el momento, a menos que ya haya sido seleccionada en ciclos anteriores. En ese caso, elegimos la fuente de alimento con el mayor número de intentos de mejora, pero tal que nunca haya sido elegida para este rol. Esto requiere mantener una lista que contiene las soluciones que ya han sido elegidas. A continuación, combinamos la fuente de alimento seleccionada con la fuente de alimento de cada abeja trabajadora, generando dos nuevos descendientes. El mejor de ellos reemplaza la fuente de alimento de la abeja trabajadora si es mejor; en ese caso, reseteamos num_trials

a cero para esta fuente de alimento; de lo contrario, la fuente de alimento original permanece en la población, incrementándose su número de intentos de mejora en una unidad. La justificación de este método es que una solución que no se mejoró después de muchos intentos de mejora probablemente sea una buena solución, y por tanto sea buena idea utilizarla como uno de los padres para generar otras soluciones. Al mismo tiempo, puede producir una diversidad razonable, ya que seleccionamos "soluciones padre" diferentes en sucesivos ciclos del algoritmo.

En el segundo enfoque (denotado e_meth2) barajamos aleatoriamente las fuentes de alimento de la población y las agrupamos en parejas para combinarlas, de una forma similar a como se hace en nuestro algoritmo genético. Por lo tanto, combinamos todas las fuentes de alimento de la población. De esta manera, esperamos que la diversidad sea mayor que en el primer método a costa de una calidad inferior.

Para combinar las soluciones utilizamos los operadores de cruce SBX y PMX, ya mencionados en anteriores secciones. Consideramos también una tercera posibilidad, que consiste en elegir aleatoriamente entre SBX y PMX cada vez que cruzamos dos soluciones. En [25] hemos demostrado que esta última propuesta obtiene mejores resultados que utilizar cada operador por separado.

Onlooker Bee Phase

Las abejas trabajadoras comparten su información con las abejas observadoras que esperan en la colmena. A continuación, las abejas observadoras escogen probabilísticamente las fuentes de alimento a las que van a ir. Una vez allí, intentan encontrar una fuente de alimento mejor en las cercanías. En particular, la probabilidad de elegir una fuente de alimento k en esta fase es:

$$prob_k = \frac{\left(\frac{1}{tard_k}\right)}{\sum_{j=1}^{SN} \frac{1}{tard_j}} \tag{4.6}$$

Hay que tener en cuenta que no existe la posibilidad de que haya divisiones por cero, ya que el algoritmo finaliza si encuentra una solución con tardiness cero.

Proponemos diferentes formas de aplicar la fase de abeja observadora, adaptadas al problema de la planificación de la carga de EVs. La idea general es adelantar un vehículo con tardiness o retrasar un vehículo sin tardiness.

La primera forma, denominada o_meth1, es una generalización del procedimiento propuesto en [25] que selecciona al azar hasta el 10 % de los EVs de la permutación. Para cada vehículo seleccionado, verifica su tardiness. Si es cero, podría retrasarse; por lo tanto, intenta intercambiarlo con todos los vehículos desde su posición hacia el final hasta que encuentre una solución mejor. Por el contrario, si el tardiness es positivo, lo intercambia con los vehículos anteriores. En cualquier caso, tan pronto como alcanza una solución mejor, ésta reemplaza a la original y resetea num_trials a cero. Si no se encuentra una solución mejor, se mantiene la solución original e incrementa num_trials en una unidad. En la generalización propuesta, mejoramos este procedimiento utilizando dos parámetros: max_improv y $step\_size$. Tan pronto como encontramos un intercambio que conduce a una solución mejor, establecemos $num\_improv \leftarrow num\_improv + 1$, y repetimos el proceso con esta nueva solución, a menos que se haya alcanzado el número máximo de mejoras max_improv. Por otro lado, utilizamos $step\_size$ para evitar probar todos los intercambios, sino solo cierto número de ellos para cada vehículo, recorriendo la permutación en pasos de

tamaño $step\_size$, en lugar de vehículo por vehículo. El parámetro max_improv permite aumentar la intensificación (por supuesto, a costa de aumentar el tiempo de cálculo), ya que en la propuesta original el algoritmo termina en cuanto se encuentra una solución que mejore, sin repetir el proceso. Por otra parte, el parámetro $step\_size$ permite que el algoritmo reduzca el tiempo de cálculo, ya que evita probar todos los posibles intercambios.

En la segunda forma de aplicar la fase de la abeja observadora (denotada o_meth2) elegimos qué EVs se adelantan o se retrasan según la estructura de la planificación. En primer lugar, seleccionamos entre planificar antes los EVs con tardiness o retrasar los que no lo tienen. Esta selección se realiza con probabilidad proporcional al número de EVs con tardiness. Por ejemplo, si el 20 % de los EVs tiene tardiness cero, entonces hay un 80 % de probabilidad de intentar retrasar los EVs y solo el 20 % de intentar adelantar los que tengan tardiness. La idea detrás de esta estrategia es que cuando hay muchos vehículos con tardiness, el retraso de algunos vehículos sin tardiness podría dar la posibilidad de planificar antes un gran número de EVs con tardiness, mientras que en situaciones con una pequeña proporción de vehículos con tardiness, podría ser mejor tratar de planificar éstos antes. Además, utilizamos un parámetro de control adicional num_steps que determina el número máximo de veces que intentamos intercambiar cada vehículo con los que lo preceden o lo suceden, y de este modo acotamos el número de máximo de intercambios y reducimos el tiempo de cálculo.

Scout Bee Phase

Cuando una solución no ha conseguido ser mejorada en un determinado número límite de intentos (denominado *limit*), se abandona y la abeja exploradora se encarga de buscar una nueva fuente de alimento. Para implementar la fase de exploración, reemplazamos todas las soluciones que han alcanzado el límite de intentos por soluciones aleatorias, y establecemos $num\_trials = 0$ para cada una de ellas.

Búsqueda local

Como procedimiento de búsqueda local proponemos aplicar una escalada simple a la solución final alcanzada por el algoritmo de la siguiente forma: iteramos sobre los EVs en el orden en que están planificados y si el EV en la posición i tiene tardiness, intentamos planificarlo antes, justo antes de cada uno de los $\lfloor i * max\_step\_perc \rfloor$ vehículos anteriores en la planificación, donde $max\_step\_perc \in [0, 1]$ es un parámetro. Si encontramos una solución mejor, sustituimos a la anterior e iniciamos de nuevo el proceso iterativo. La búsqueda local finaliza cuando la solución no mejora en un ciclo iterativo completo. El parámetro max_step_perc recibe un valor pequeño por razones de eficiencia y para introducir cambios razonablemente pequeños en las soluciones vecinas. Finalmente, hay que destacar que el número máximo de EVs con los que probamos a intercambiar cada vehículo depende de su posición en la permutación.

4.2.6. Algoritmo genético difuso

En [27] proponemos un algoritmo genético para resolver el problema de planificación de EVs con tiempos de carga difusos, un Algoritmo Genético

Difuso (en inglés fuzzy Genetic Algorithm, fGA). Este fGA está basado en el propuesto en [23], y descrito en la Sección 4.2.2 de esta tesis, para la versión determinista del problema. La principal diferencia es la utilización del generador de planificaciones para el problema difuso f-SGS, descrito en la Sección 4.1.3.

Capítulo 5

Estudio experimental

Todos los algoritmos propuestos en esta tesis están implementados en C++ y los estudios experimentales se han llevado a cabo en un servidor Xeon E5520 con un sistema operativo Linux (SL6.0). Dada la naturaleza estocástica de este tipo de algoritmos, todas experimentos se han ejecutado 30 veces para obtener resultados estadísticamente significativos.

5.1. El problema determinista

5.1.1. Banco de instancias

En las diferentes publicaciones de esta tesis hemos considerado un conjunto de instancias propuesto en [38]. Las instancias corresponden a una estación de carga con 180 plazas de aparcamiento, y los perfiles de tiempos de llegada, tiempos de carga e instantes de recogida se generan para simular distintos tipos de comportamiento en la llegada de usuarios al garaje. El horizonte temporal es de un día y se consideran tres escenarios diferentes: el escenario 1 representa un día de semana normal, con vehículos que llegan durante todo el día, pero con algunos picos de llegada. El 10 % de los vehículos llegan uniformemente a lo largo de todo el día, el 20 % llegan alrededor de las 8:30, el 10 % alrededor de las 12:00, el 50 % alrededor de las 19:30 y el 10 % restante alrededor de las 22:00. Para simular estas condiciones se han utilizado las distribuciones de probabilidad de la Tabla 5.1 (escenario 1). Por otro lado, los escenarios 2 y 3 representan una situación más extrema donde la mayoría de los vehículos llegan casi al mismo tiempo. La diferencia entre los escenarios 2 y 3 es que en el último los instantes de recogida son más ajustados. La Tabla 5.1 (escenarios 2 y 3) muestra las distribuciones de probabilidad utilizadas para generar los tiempos de llegada de estas instancias.

Con respecto a la carga de las baterías cuando los vehículos llegan a la estación de carga, en los tres escenarios se considera que el 10 % tienen aproximadamente el 80 % de su capacidad, el 30 % aproximadamente el 50 %, el 30 % aproximadamente el 35 % y el 30 % restante aproximadamente el 12 %. Los tiempos de carga se calculan suponiendo que todos los vehículos requieren una carga del 100 % de su capacidad, 23 kW, y que la velocidad de carga es de 2.3 kWh. Los porcentajes y distribuciones utilizadas se pueden consultar en la Tabla 5.2.

Hay dos tipos de instancias con diferentes distribuciones de vehículos en las líneas. En las instancias de *Tipo* 1 llegan 60 vehículos a cada línea a lo largo del día,

Tabla 5.1: Resumen de los porcentajes y distribuciones de probabilidad utilizadas para generar los tiempos de llegada.

Escenario 1		Escenarios 2 y 3	
% EVs	Tiempo de llegada	% EVs	Tiempo de llegada
10	$U(0, 1440)$	20	$U(0, 1440)$
20	$N(510, 15)$	80	$N(870, 30)$
10	$N(720, 15)$		
50	$N(1170, 15)$		
10	$N(1350, 15)$		

Tabla 5.2: Resumen de los porcentajes y distribuciones de probabilidad utilizadas para generar las cargas iniciales y los instantes de recogida.

% EVs	Carga inicial (%)	Inst. recogida (esc. 1 y 2)	Inst. recogida (esc. 3)
10	$N(80, 10)$	$N(240, 120)$	$N(120, 60)$
30	$N(50, 15)$	$N(360, 120)$	$N(300, 60)$
30	$N(35, 7.5)$	$N(480, 120)$	$N(360, 60)$
30	$N(12, 6)$	$N(660, 120)$	$N(540, 60)$

mientras que en las instancias de *Tipo* 2 llegan 108 vehículos a la línea L_1 (60 %), 54 a L_2 (30 %) y 18 a L_3 (10 %). Claramente las instancias de *Tipo* 2 promueven situaciones de desequilibrio entre las líneas, y por lo tanto puede ser más difícil obtener una solución satisfactoria para ellas. Se consideran 4 valores diferentes para el parámetro de desequilibrio Δ ($0.2, 0.4, 0.6$ y 0.8), y tres valores para el número máximo de vehículos N que se pueden cargar al mismo tiempo en una línea dada ($20, 30$ y 40).

Para cada una de las 72 combinaciones posibles de tuplas (escenario, $Tipo$, Δ, N), hay 30 instancias, por lo tanto tenemos un total de 2160 instancias.

5.1.2. Visualizador gráfico de planificaciones

Hemos desarrollado un programa de visualización gráfica de las planificaciones generadas para instancias del problema determinista, con el objetivo de ayudarnos a mejorar los algoritmos diseñados y detectar errores.

La Figura 5.1 muestra un ejemplo de planificación de una instancia del escenario 1, *Tipo* 1, $N = 20$ y $\Delta = 0.4$. El eje X corresponde al tiempo, mientras que el eje Y representa, en las tres primeras gráficas, la ocupación de cada línea en cada instante de tiempo (en azul). Los intervalos de carga de cada vehículo estan representados por rectángulos de color verde, si el vehículo termina su carga a tiempo, y en amarillo y rojo en caso contrario (utilizamos el color rojo para representar el tiempo que excede su due date, que corresponde al tardiness del vehículo). También se muestra en la cuarta gráfica el nivel de desbalanceo entre las líneas en cada instante. Cuando el nivel de desbalanceo alcanza 0.4 significa que hay 8 vehículos de diferencia entre dos de las líneas, que es el máximo desbalanceo permitido.

En las Figuras 5.2, 5.3 y 5.4 podemos ver ejemplos adicionales de planificaciones para instancias del escenario 1, con distintos valores de los parámetros *Tipo*, N y Δ. Se aprecia que en las instancias de *Tipo* 2 el nivel de desbalanceo casi siempre es máximo debido a la carga desigual entre las líneas. Esto provoca que los vehículos de

las líneas L_1 y L_2 se tengan que retrasar. También podemos observar que, a medida que los parámetros N y Δ aumentan, es más sencillo obtener una planificación con un menor tardiness.

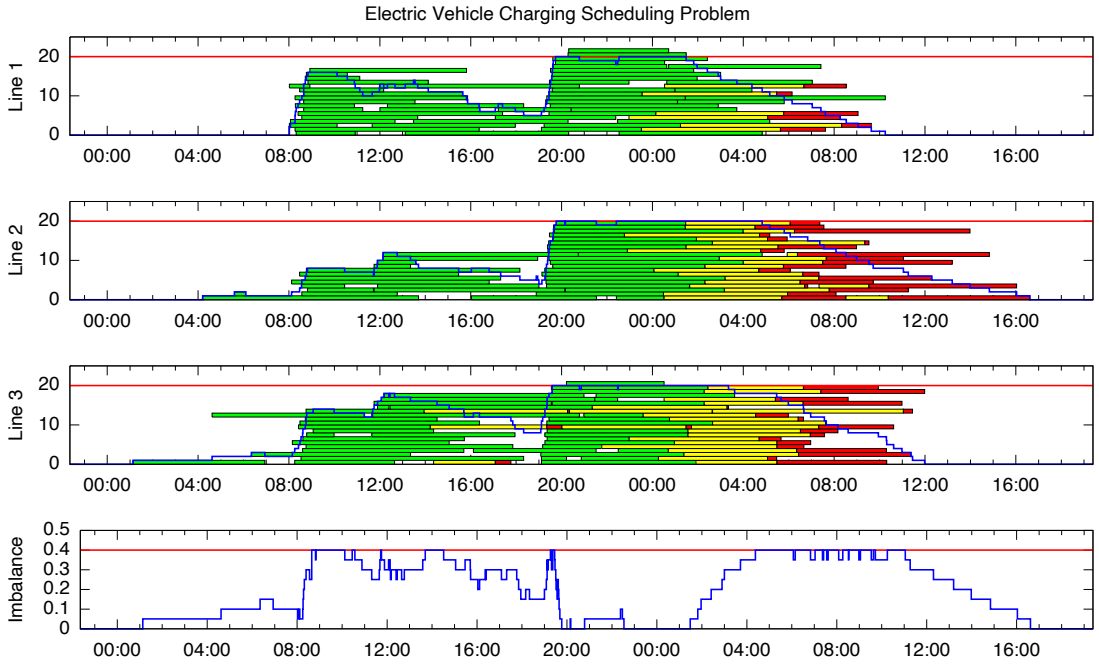
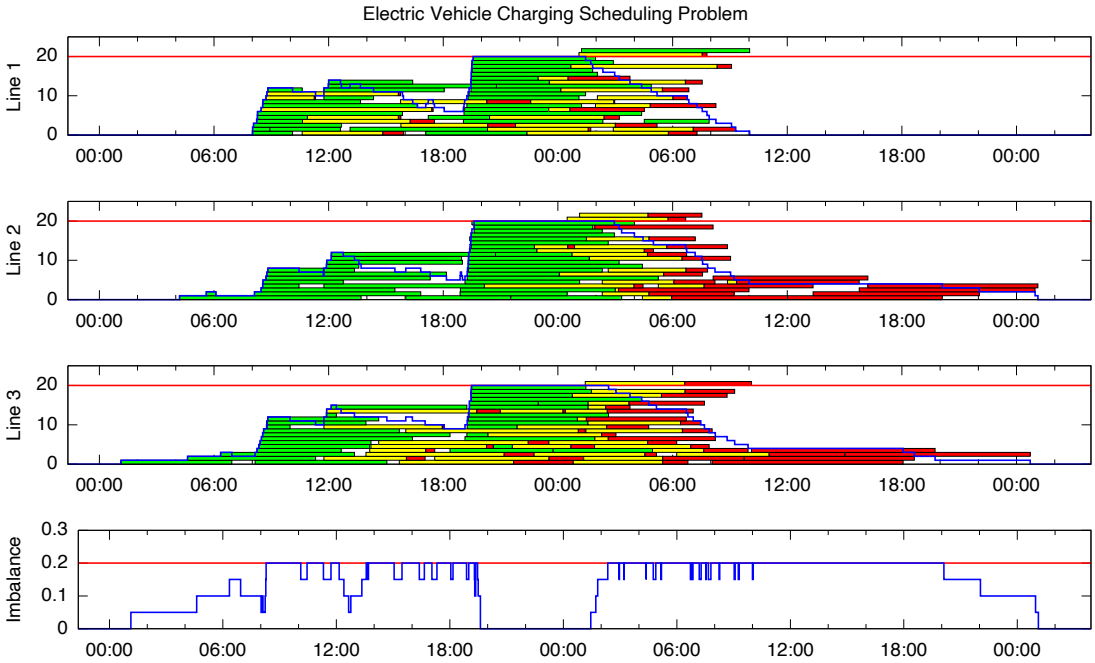
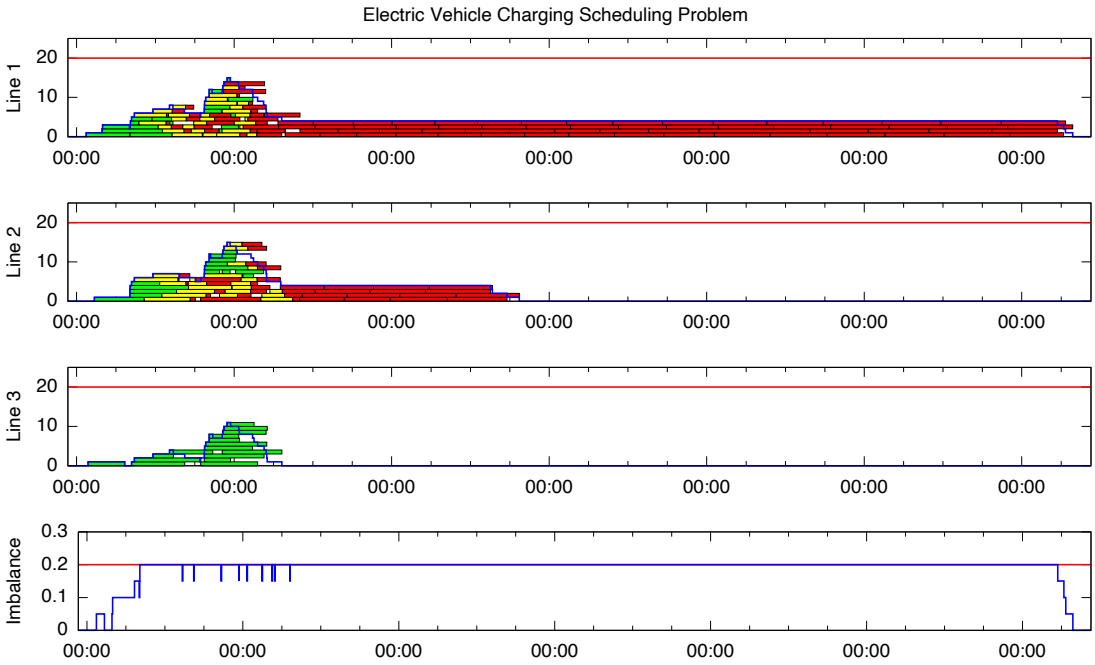


Figura 5.1: Ejemplo de planificación de una instancia de un problema.

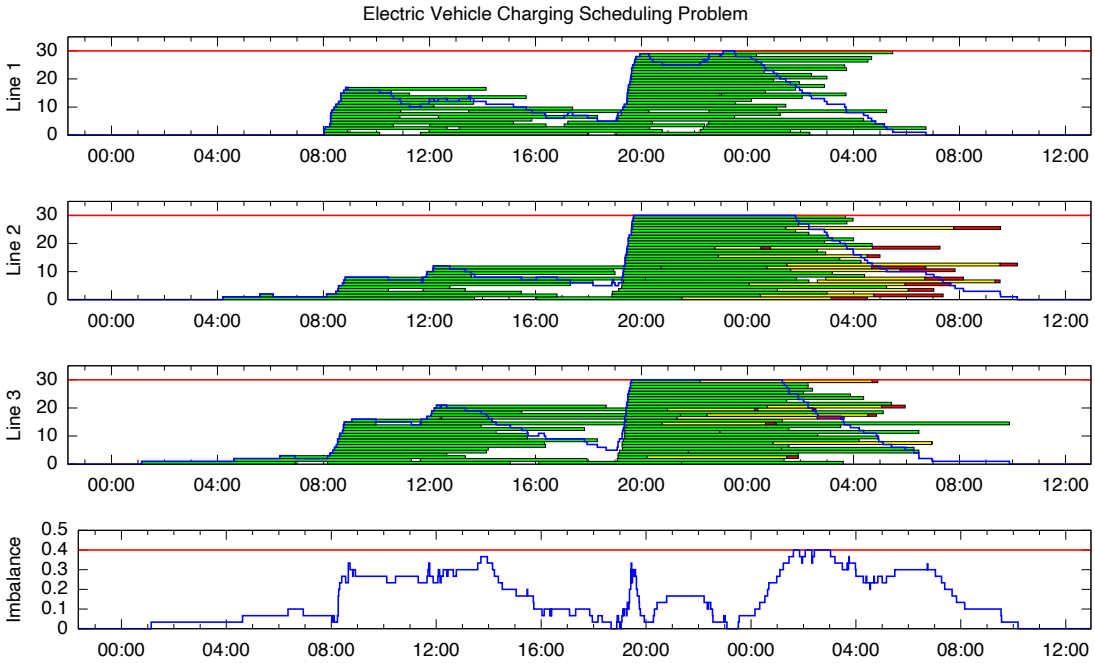


(a) Instancia de *Tipo* 1.

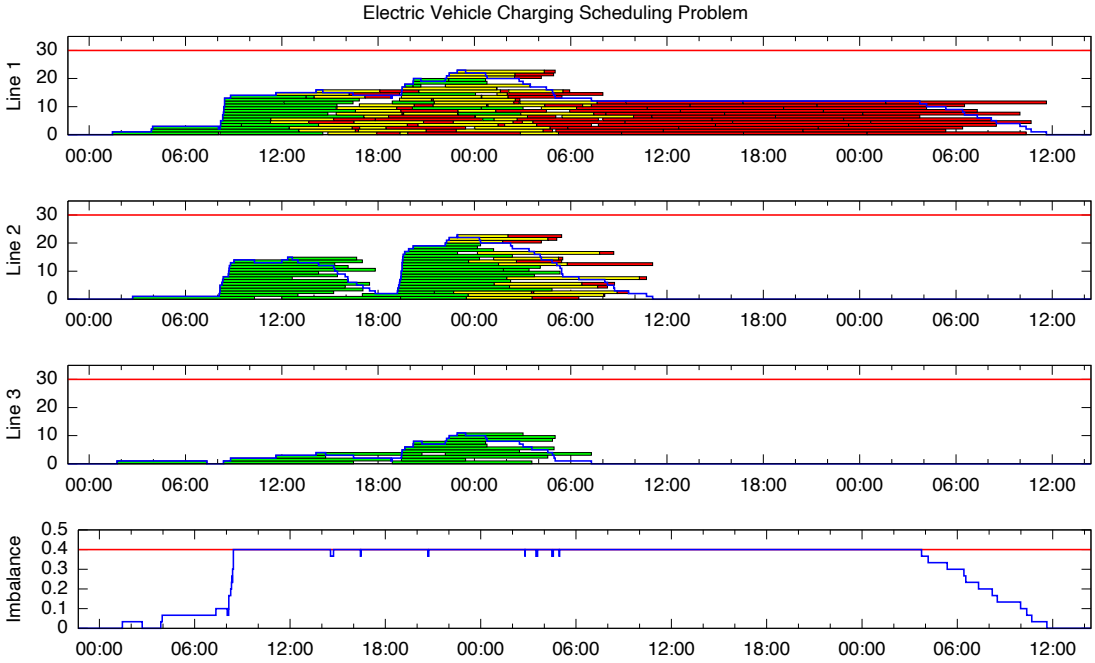


(b) Instancia de *Tipo* 2.

Figura 5.2: Ejemplos de instancias del escenario 1 con $N = 20$ y $\Delta = 0.2$.

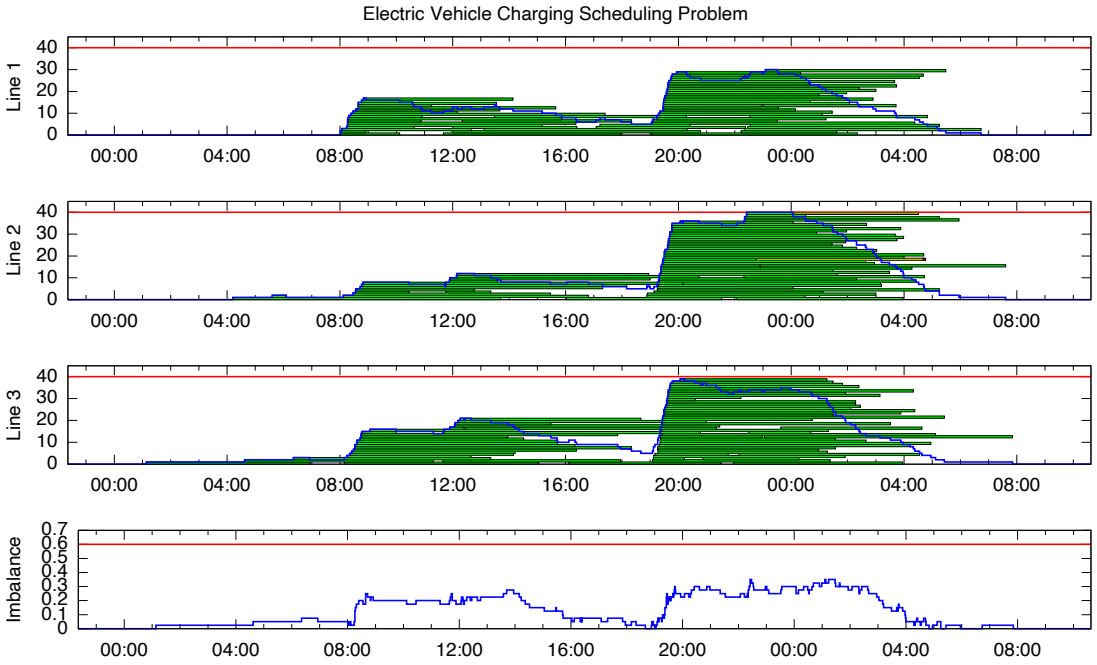


(a) Instancia de *Tipo* 1.

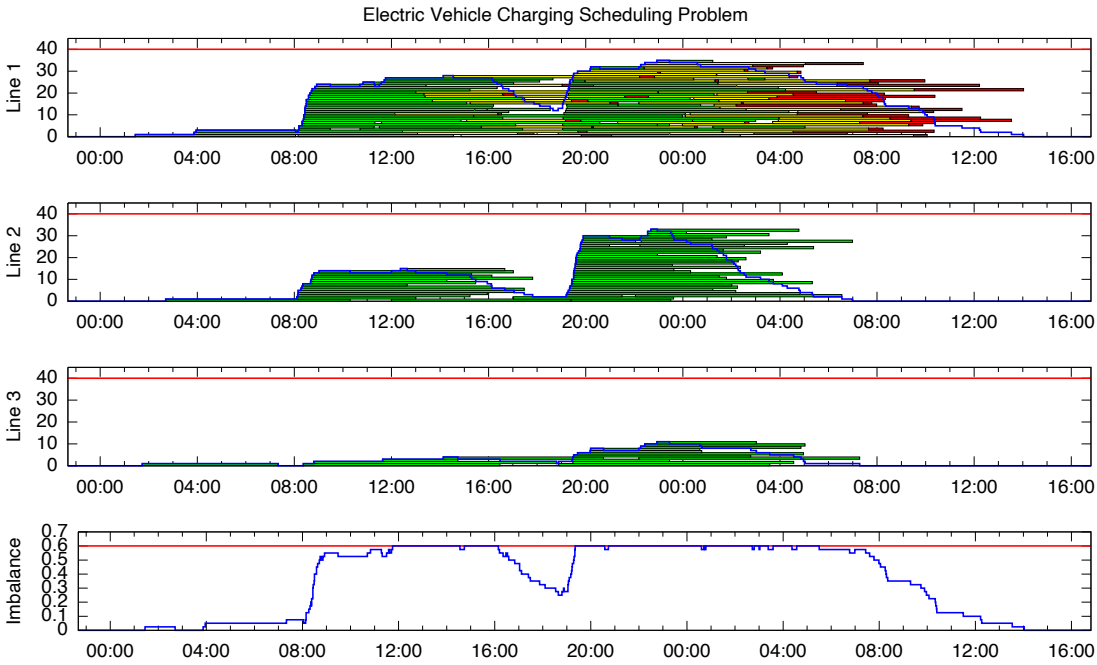


(b) Instancia de *Tipo* 2.

Figura 5.3: Ejemplos de instancias del escenario 1 con $N = 30$ y $\Delta = 0.4$.



(a) Instancia de *Tipo* 1.



(b) Instancia de *Tipo* 2.

Figura 5.4: Ejemplos de instancias del escenario 1 con $N = 40$ y $\Delta = 0.6$.

5.1.3. Resultados

Los análisis de los resultados experimentales de todas las metaheurísticas propuestas se han realizado en dos fases. En la primera se define la mejor configuración buscando tanto los mejores valores de los parámetros de cada algoritmo como los operadores de entre los propuestos que ofrecen un mejor rendimiento. Para esta fase se ha utilizado un conjunto reducido de instancias, concretamente la primera instancia de cada grupo del escenario 1. La segunda fase tiene como objetivo contrastar los resultados de cada metaheurística propuesta con lo que en ese momento sea el estado de arte. Se incluyen tests estadísticos para confirmar, en cada caso, si las diferencias entre resultados son significativas.

Resultados del GA

En [23] hemos realizado un estudio experimental para evaluar el GA descrito en la Sección 4.2.2 y compararlo con los resultados publicados en el estado del arte. Tras un estudio preliminar para determinar algunos de los parámetros de entrada hemos decidido, para obtener tiempos de ejecución razonables, utilizar un tamaño de población de 200 cromosomas y una condición de parada de 25 generaciones consecutivas sin mejorar la mejor solución encontrada hasta el momento.

A continuación hemos experimentado con diferentes valores para el resto de parámetros, es decir, la probabilidad de cruce P_c, la probabilidad de mutación P_m y el tamaño del torneo $tSize$ que se utiliza en las reglas de prioridad con las que creamos la población inicial. La Tabla 5.3 muestra el resumen de los valores probados, indicando en negrita la mejor configuración encontrada.

Tabla 5.3: Valores probados para elegir los parámetros del GA. En negrita los valores de la mejor configuración encontrada.

Parámetro	Valores probados
P_c	0.6, **0.8**, 1
P_m	0, **0.1**, 0.3
$tSize$	4, **8**, 12, 16

La configuración descrita genera unos patrones de convergencia razonables, como se muestra en la Figura 5.5, donde podemos ver la evolución del tardiness del mejor individuo y el tardiness medio de la población para una ejecución en una instancia del escenario 1, $Tipo$ 1, $N = 20$ y $\Delta = 0.2$.

A continuación comparamos los operadores de cruce LEX, SBX y 1PX. En la Tabla 5.4 mostramos los resultados de las versiones estática y dinámica del problema agrupados por tipo de instancia, para un conjunto de 24 instancias del escenario 1. Como podemos ver, el operador SBX siempre obtiene los mejores resultados.

Para analizar si las diferencias son estadísticamente significativas hemos realizado en primer lugar un test Shapiro-Wilk para comprobar la normalidad de los datos y, descartada la normalidad, hemos utilizado un test de Wilcoxon para muestras pareadas que nos ha permitido descartar la hipótesis de que la diferencia entre los valores medios obtenidos con el operador SBX y con el resto de operadores de cruce propuestos sea menor de cero.

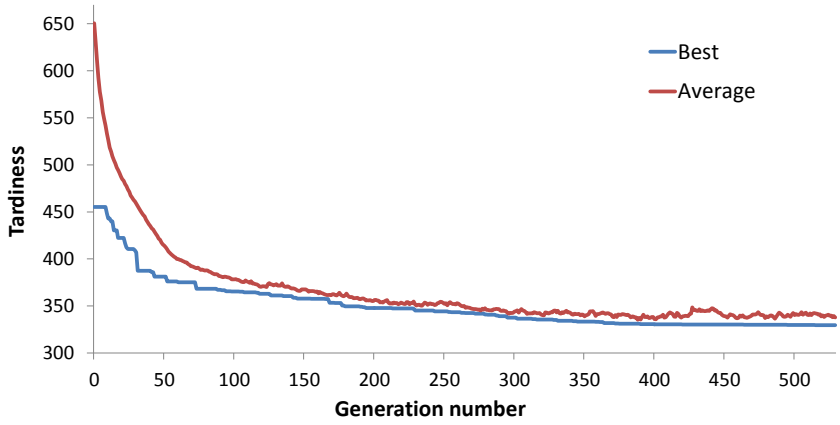


Figura 5.5: Evolución del mejor tardiness y del tardiness medio de la población de una ejecución del GA.

Tabla 5.4: Comparación entre los operadores de cruce propuestos para un conjunto de 24 instancias. Los resultados muestran el tardiness medio (en horas) de 30 ejecuciones agrupadas por tipo de instancia.

Inst.	Estático			Dinámico		
	LEX	SBX	1PX	LEX	SBX	1PX
Tipo 1	56.34	**55.27**	55.67	73.24	**71.79**	72.11
Tipo 2	1157	**1113**	1118	1129	**1119**	1121

Una vez determinado el mejor operador de cruce, realizamos una comparación de nuestro GA, utilizando el operador de cruce SBX, con el algoritmo Problem Decomposition (PD) propuesto en [37] y que, por lo que sabemos, es el único algoritmo existente en la literatura que resuelve exactamente el mismo problema. En este caso hemos considerado las 720 instancias del escenario 1, tanto para evaluar la versión estática como la dinámica del problema. En la Tabla 5.5 mostramos los resultados obtenidos. Cada grupo está compuesto por 30 instancias y en la tabla mostramos la suma de sus tardiness. Debemos comparar GA con PD en la versión dinámica del problema, que es la única que se considera en [37]. Hemos realizado tests estadísticos similares a los descritos en párrafos anteriores para confirmar que las diferencias entre los valores medios de los tardiness obtenidos por GA y PD en las instancias consideradas son estadísticamente significativas, y por lo tanto podemos concluir que nuestro algoritmo obtiene resultados mejores en tardiness. Tal y como esperábamos, los resultados en la versión estática del problema son mucho mejores que los de la versión dinámica. Podemos concluir que el GA explota toda la información adicional sobre los tiempos de llegada, los tiempos de carga y los instantes de recogida para llegar a soluciones mejores.

Sin embargo debemos tener en cuenta el tiempo de ejecución, ya que PD, al ser un método más simple basado en reglas de prioridad, es mucho más rápido, y sus ejecuciones son siempre inferiores a 0.012 segundos. El tiempo promedio del GA en el problema estático es de 93 segundos por ejecución, aunque el tiempo varía significativamente en función de los parámetros de la instancia. De hecho, las instancias con $\Delta = 0.2$ generalmente tardan más tiempo, ya que las restricciones de desequilibrio son mucho más restrictivas y el problema es más difícil de resolver. En

Tabla 5.5: Comparación del GA propuesto con el algoritmo PD del estado del arte. Cada valor corresponde a la suma de los tardiness (en horas) de las 30 instancias de cada grupo.

N	Δ	PD	GA (Estático)		GA (Dinámico)	
			Mejor	Media	Mejor	Media
Instancias *Tipo* 1						
20	0.2	7720	5250	5442	6744	7142
	0.4	4240	2616	2680	3844	3977
	0.6	3849	2260	2300	3494	3569
	0.8	3807	2204	2239	3435	3502
30	0.2	1782	931	997	1262	1412
	0.4	490	80	92	343	375
	0.6	458	37	50	302	319
	0.8	458	35	49	300	317
40	0.2	646	337	364	464	511
	0.4	28	0	0	4	6
	0.6	9	0	0	3	3
	0.8	9	0	0	3	3
Media		1958	1146	1184	1683	1761
Instancias *Tipo* 2						
20	0.2	127614	123167	124380	123746	124599
	0.4	46254	44675	45263	45049	45461
	0.6	23008	20957	21206	21771	22075
	0.8	14808	12888	13031	14107	14337
30	0.2	72460	70304	71129	70825	71462
	0.4	21427	20378	20630	21043	21321
	0.6	8079	7093	7188	7837	8007
	0.8	4501	3551	3607	4325	4408
40	0.2	46096	44618	45216	45031	45455
	0.4	10932	9883	10011	10593	10799
	0.6	3520	2870	2917	3432	3518
	0.8	1659	888	923	1524	1569
Media		31696	30106	30458	30773	31084

el problema dinámico, el tiempo promedio es de 325 segundos. Sin embargo, este número representa el tiempo total de ejecución a lo largo del período de 24 horas. Como el algoritmo se ejecuta cada dos minutos (si ha llegado algún vehículo a la estación de carga), se ejecuta un gran número de veces durante el período de 24 horas y, por lo tanto, el tiempo de ejecución de cada subproblema es mucho más reducido, siendo el tiempo del subproblema que más tiempo ha consumido 2.75 segundos, por lo que podemos concluir que el GA es apropiado en un entorno del mundo real.

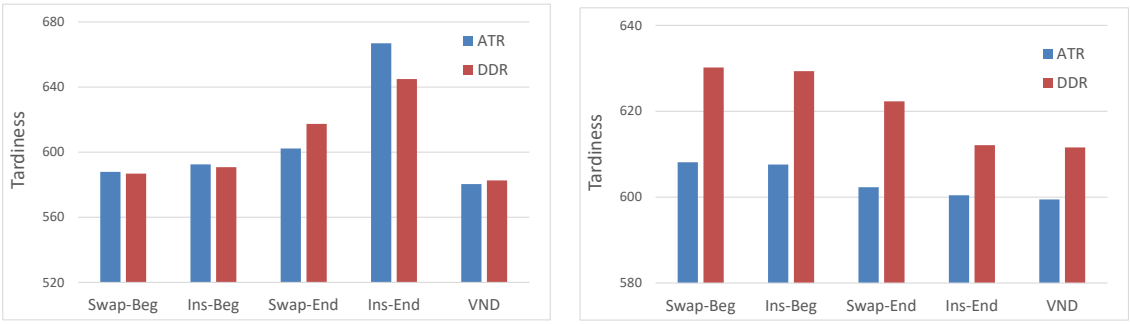
Resultados del GRASP y del MA

En [24] hemos realizado un estudio experimental para evaluar los algoritmos GRASP y MA descritos en las Secciones 4.2.3 y 4.2.4, y compararlos con los resultados publicados en el estado del arte. De nuevo, el primer paso es hacer un estudio paramétrico con un grupo reducido de instancias para ajustar el valor de los parámetros de entrada de los dos algoritmos.

Comenzamos buscando la mejor configuración del algoritmo GRASP para la versión estática del problema. Para ello analizamos el comportamiento del algoritmo considerando las reglas de prioridad ATR y DDR con diferentes tamaños de torneo, utilizando los algoritmos de generación de planificaciones b-SGS y a-SGS, y probando las diferentes búsquedas locales propuestas. Establecemos el número de reinicios $NRestarts$ en 25, ya que este valor hace que el algoritmo consuma un tiempo de ejecución razonable. El tamaño del torneo que logró el mejor tardiness promedio es 2, tanto para la regla de prioridad ATR como para la DDR. A continuación evaluamos el rendimiento de los algoritmos de generación de planificaciones. En la Tabla 5.6 mostramos una comparación entre el algoritmo b-SGS y el a-SGS. Para obtener tiempos de ejecución similares, hemos establecido el valor de $NRestarts$ en 50 para el planificador estándar y en 25 para el mejorado. Finalmente comparamos las cuatro estrategias de búsqueda local propuestas, es decir, intercambio o inserción iterando desde el principio o el fin de la permutación. En la Figura 5.6(a) mostramos el tardiness promedio (en horas) obtenido en todos los casos considerados. El orden de mejor a peor vecindad es *Swap-Beg*, *Ins-Beg*, *Swap-End* y finalmente *Ins-End*. También mostramos los resultados de la VND utilizando las dos vecindades que mejor rendimiento han dado, es decir, $N_1 = Swap\text{-}Beg$ y $N_2 = Ins\text{-}Beg$. Para obtener tiempos de ejecución similares, establecemos el valor de $NRestarts$ en 25 cuando se utiliza una sola estructura de vecindad y en 20 cuando se utiliza VND. Se puede ver que los mejores resultados son los de la búsqueda local VND. Por lo tanto, concluimos que la mejor configuración para el algoritmo GRASP en el problema estático es utilizar la regla de prioridad ATR con un tamaño de torneo de 2, el planificador a-SGS y la búsqueda local VND con las vecindades $N_1 = Swap\text{-}Beg$ y $N_2 = Ins\text{-}Beg$.

Tabla 5.6: Comparación de los algoritmos de generación de planificaciones. Los valores representan el tardiness promedio (en horas) agrupados por tipo de instancia.

Inst.	ATR		DDR	
	b-SGS	a-SGS	b-SGS	a-SGS
Tipo 1	58.3	**55.2**	57.8	**55.2**
Tipo 2	1140.5	**1120.6**	1138.2	**1118.4**



(a) Problema estático.

(b) Problema dinámico.

Figura 5.6: Comparación de las diferentes estrategias de búsqueda local.

Como la mejor configuración para las versiones estática y dinámica del problema puede diferir, repetimos el mismo proceso para buscar la mejor configuración para

la versión dinámica del problema. Comenzamos seleccionando el tamaño de torneo más apropiado para cada regla de prioridad. Nuevamente, el tamaño del torneo que alcanza el mejor tardiness promedio es 2 para ambas reglas. Posteriormente, hemos confirmado que el algoritmo a-SGS también mejora los resultados obtenidos por b-SGS. En la Figura 5.6(b) mostramos una comparación de las estrategias de búsqueda local. En este caso, el orden de las vecindades de mejor a peor es *Ins-End*, *Swap-End*, *Ins-Beg* y finalmente *Swap-Beg*. La configuración que logra los mejores valores promedio de tardiness es una VND con las dos mejores vecindades: N_1 = *Ins-End* y N_2 = *Swap-End*. En resumen, la mejor configuración para la versión dinámica del problema es utilizar la regla de prioridad ATR con un tamaño de torneo de 2, el planificador a-SGS y un enfoque VND con las vecindades N_1 = *Ins-End* y N_2 = *Swap-End*. Por lo tanto, el único cambio con respecto al problema estático son las estructuras de vecindad elegidas.

Como nuestro MA está basado en el GA propuesto en [23], tomamos como configuración base los parámetros establecidos en esa publicación: un tamaño de población PS de 200 cromosomas, la condición de parada fijada en 25 generaciones consecutivas sin mejorar la mejor solución encontrada, probabilidad de cruce P_c = 0.8, probabilidad de mutación $P_m = 0.1$ para cada línea, y tamaño de torneo $tSize$ de 8 para las reglas de prioridad utilizadas para crear la población inicial. Para la búsqueda local, proponemos utilizar la mejor configuración encontrada para el algoritmo GRASP. Esta configuración da como resultado patrones de convergencia razonables, como se muestra en la Figura 5.7, que detalla la evolución del mejor tardiness y del tardiness promedio de la población para la ejecución de una instancia del escenario 1, $Tipo$ 1, $N = 20$ y $\Delta = 0.2$. Hay que destacar la brusca bajada del tardiness en la última generación, que es debida a la aplicación de la búsqueda local. Como el MA presentado tiene algunas características adicionales con respecto al algoritmo genético usado como referencia, debemos asegurarnos de que la configuración base elegida sea la mejor posible. Para ello, hacemos pruebas modificando la probabilidad de cruce (P_c), la probabilidad de mutación (P_m) y el tamaño de la población con respecto a la configuración base. La Figura 5.8 muestra los resultados obtenidos en las instancias de $Tipo$ 1 y $Tipo$ 2 del problema estático; esto nos lleva a fijar una probabilidad de cruce $P_c = 0.8$, una probabilidad de mutación $P_m = 0.1$ y un tamaño de población $PS = 200$. Otra decisión relevante es elegir cuántos cromosomas deberíamos mejorar con la búsqueda local. La Tabla 5.7 muestra los resultados obtenidos en el problema estático aplicando la búsqueda local a un porcentaje diferente de soluciones de la población final. Obviamente, obtenemos los mejores resultados al mejorar todos los cromosomas, pero el tiempo de ejecución aumenta enormemente y no compensa la leve mejora en los valores del tardiness. Por estas razones, proponemos el 5 % como la mejor alternativa. Los resultados en el problema dinámico son similares, aunque las diferencias en los tiempos de ejecución son menores debido al hecho de que los subproblemas suelen ser pequeños y, por lo tanto, la búsqueda local es en promedio mucho más rápida.

Las Tablas 5.8, 5.9 y 5.10 muestran la comparación entre la mejor configuración de los algoritmos GRASP y MA, y el estado del arte para los de escenarios 1, 2 y 3, respectivamente. Los resultados están agrupados en conjuntos de 30 instancias que dependen de los parámetros de la instancia ($Tipo$, N y Δ). Los valores corresponden a la suma del tardiness de las 30 instancias de cada grupo. La columna GA muestra los valores promedio obtenidos por el algoritmo genético propuesto en [23] para

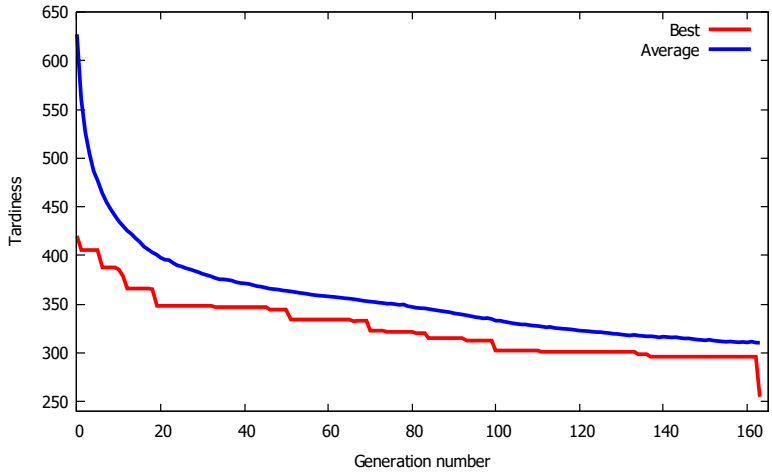
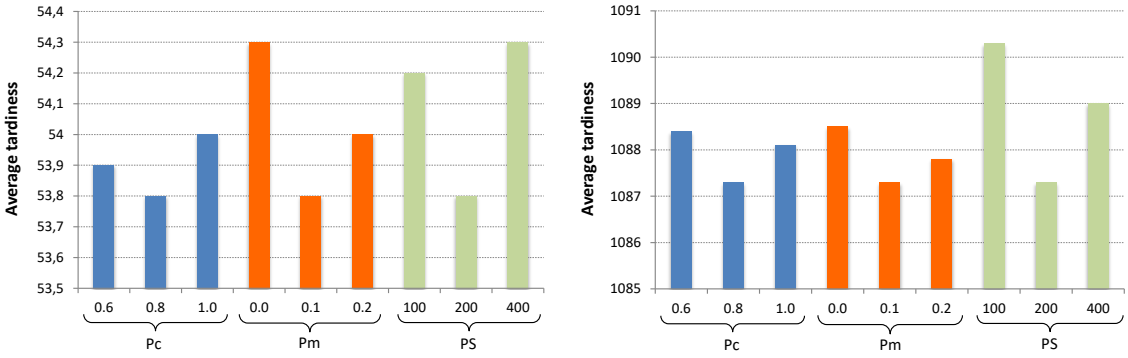


Figura 5.7: Evolución del mejor tardiness y del tardiness medio de la población de una ejecución del MA. Nótese la aplicación de la búsqueda local en la brusca bajada del tardiness en la última generación.



(a) Instancias de Tipo 1.

(b) Instancias de Tipo 2.

Figura 5.8: Comparación de las diferentes configuraciones del MA.

Tabla 5.7: Comparación del tardiness y tiempos de ejecución al variar el porcentaje de cromosomas a los que se aplica búsqueda local en la población final del MA. Los resultados son las medias agrupadas por tipo de instancia.

Inst.	0 %		5 %		100 %	
	$tard_S$	T(s)	$tard_S$	T(s)	$tard_S$	T(s)
Tipo 1	58.0	12.9	53.8	28.1	53.0	177.2
Tipo 2	1092.9	38.6	1087.3	80.5	1085.9	330.7

el escenario 1. La columna PD muestra los valores del tardiness obtenidos por el algoritmo PD propuesto en [38] para el problema dinámico en los tres escenarios. Las columnas GRASP y MA muestran los resultados promedio de nuestros algoritmos. Marcamos en negrita el mejor valor para cada versión del problema.

Vemos de nuevo que los resultados en la versión estática del problema son mucho mejores que los de la versión dinámica. En particular, para el algoritmo GRASP, la solución promedio en el problema estático es mejor que la del problema dinámico en 2037 instancias, igual en 72 y peor en 51, mientras que en el caso del MA es mejor

Tabla 5.8: Comparación con el estado del arte en las instancias del escenario 1. Cada valor representa la suma del tardiness (en horas) de las 30 instancias de cada grupo.

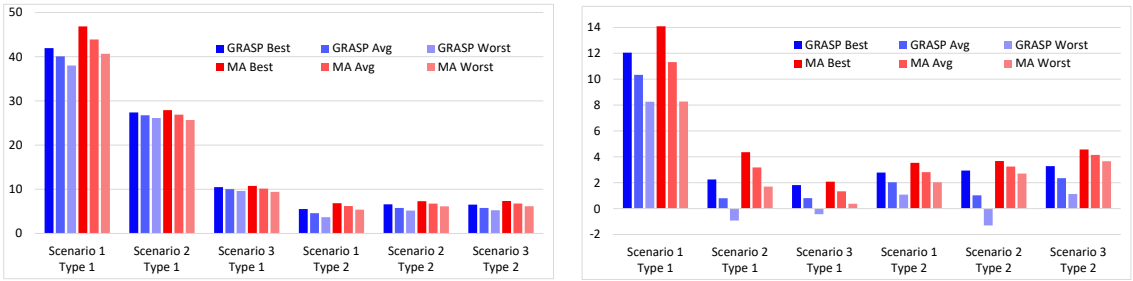
N	Δ	Problema Estático			Problema Dinámico			
		GA [23]	GRASP	MA	PD [38]	GA [23]	GRASP	MA
Instancias *Tipo* 1								
20	0.2	5442.3	5348.9	**5210.8**	8386.3	7141.6	7182.9	**6981.2**
	0.4	2680.1	2651.3	**2509.2**	4120.4	3976.7	3918.2	**3884.2**
	0.6	2299.7	2320.3	**2252.7**	3670.6	3568.8	**3556.1**	3556.7
	0.8	2238.7	2332.7	**2199.0**	3590.9	3501.9	**3496.6**	3506.3
30	0.2	996.9	983.2	**729.5**	1959.3	**1411.7**	1481.3	1444.8
	0.4	92.1	82.2	**52.5**	421.2	374.7	**355.8**	367.6
	0.6	50.0	36.2	**34.3**	347.9	318.9	**306.5**	316.2
	0.8	49.2	38.1	**33.8**	347.6	316.8	**304.5**	314.7
40	0.2	364.1	350.8	**216.9**	735.0	**511.0**	546.6	541.6
	0.4	0.4	**0.0**	**0.0**	14.0	6.4	**5.7**	7.7
	0.6	**0.0**	**0.0**	**0.0**	**3.4**	**3.4**	**3.4**	**3.4**
	0.8	**0.0**	**0.0**	**0.0**	**3.4**	**3.4**	**3.4**	**3.4**
Media		1184.5	1178.6	**1103.2**	1966.7	1761.3	1763.4	**1744.0**
Instancias *Tipo* 2								
20	0.2	124380.0	124565.0	**122615.0**	128185.0	124599.3	125226.8	**124075.0**
	0.4	45263.1	44675.0	**43991.1**	46319.3	45461.3	45407.4	**45127.2**
	0.6	21205.6	21025.6	**20526.9**	22966.8	22074.6	22089.9	**21822.6**
	0.8	13031.0	13053.2	**12762.8**	14573.1	14337.1	14392.7	**14209.9**
30	0.2	71129.0	70682.9	**69610.6**	72860.8	71462.5	71371.7	**70894.5**
	0.4	20629.5	20456.0	**20031.7**	21479.9	21321.0	21311.3	**21130.3**
	0.6	7188.0	7204.1	**7030.5**	8088.9	8006.8	7984.5	**7921.5**
	0.8	3607.7	3567.8	**3527.9**	4486.3	4407.8	4391.4	**4389.3**
40	0.2	45216.0	44658.3	**43981.8**	46135.4	45455.4	45416.1	**45139.2**
	0.4	10010.6	10009.9	**9760.7**	10869.3	10799.0	10802.5	**10654.6**
	0.6	2916.8	2886.9	**2851.1**	3599.1	3517.8	**3505.9**	3515.6
	0.8	922.8	892.4	**872.3**	1635.5	1568.8	**1554.6**	1572.1
Media		30458.3	30306.4	**29796.9**	31766.6	31084.3	31121.2	**30871.0**

en 2077 instancias, igual en 72 instancias y peor en 11 instancias. Con respecto a la comparación entre nuestros resultados para la versión dinámica y los de PD [38], el resultado promedio de GRASP es mejor en el 71 % de instancias, mientras que el resultado promedio de MA es mejor en el 85 % de instancias. También podemos observar que tanto GRASP como MA obtienen generalmente mejores resultados que el GA propuesto en [23]. Finalmente, si comparamos los resultados de GRASP con los de MA, vemos que MA obtiene un mejor resultado promedio en el 83 % de las instancias para el problema estático y en el 61 % de las instancias para el problema dinámico.

Las Figuras 5.9(a) y 5.9(b) muestran el porcentaje de mejora promedio de nuestros métodos con respecto a los resultados publicados en [38], en el problema estático y dinámico respectivamente. Las mayores mejoras son las obtenidas en instancias de *Tipo* 1 considerando el escenario 1. Además, las mejoras en la versión estática del problema (más del 40 % en algunos casos) son mucho mayores que las obtenidas en la versión dinámica (hasta el 14 %). También es destacable la pequeña diferencia entre nuestros mejores y peores resultados, lo que demuestra la robustez de los métodos.

Tabla 5.9: Comparación con el estado del arte en las instancias del escenario 2. Cada valor representa la suma del tardiness (en horas) de las 30 instancias de cada grupo.

N	Δ	Problema Estático		Problema Dinámico		
		GRASP	MA	PD [38]	GRASP	MA
Instancias *Tipo* 1						
20	0.2	**14015.5**	14411.5	16886.0	16772.7	**16176.3**
	0.4	11798.4	**11703.6**	14131.2	14429.1	**13915.9**
	0.6	11392.1	**11328.3**	13648.3	13941.3	**13559.1**
	0.8	11300.1	**11242.1**	13547.9	13817.9	**13492.7**
30	0.2	3936.4	**3920.6**	6394.7	5995.3	**5821.5**
	0.4	2745.4	**2726.2**	4824.3	4716.3	**4688.4**
	0.6	2621.1	**2592.8**	4668.6	**4546.4**	4560.3
	0.8	2627.7	**2591.4**	4672.6	**4544.0**	4559.0
40	0.2	820.2	**638.0**	1951.4	1626.4	**1593.5**
	0.4	190.6	**186.4**	1212.7	**1096.2**	1102.0
	0.6	174.9	**172.2**	1210.7	**1098.8**	1103.3
	0.8	177.0	**171.9**	1210.6	**1099.2**	1102.7
Media		5149.9	**5140.4**	7029.9	6973.6	**6806.2**
Instancias *Tipo* 2						
20	0.2	138943.0	**137204.0**	143883.0	142949.9	**139192.0**
	0.4	58014.0	**57453.4**	62246.2	61157.7	**59194.0**
	0.6	34846.1	**34548.2**	39869.4	36887.1	**36298.4**
	0.8	27861.8	**27735.1**	30887.2	29955.0	**29791.0**
30	0.2	84357.5	**82970.0**	86392.0	87569.1	**84470.0**
	0.4	32928.7	**32714.9**	34475.4	34501.4	**33936.8**
	0.6	17195.6	**17077.3**	19355.7	18723.6	**18709.8**
	0.8	12405.9	**12364.5**	14375.1	**14020.1**	14201.7
40	0.2	58178.9	**57652.0**	59775.0	60705.5	**58714.1**
	0.4	20910.2	**20786.9**	22254.0	22115.0	**21984.2**
	0.6	9637.2	**9582.4**	10991.3	**10691.2**	10815.9
	0.8	5816.4	**5806.0**	7260.5	**7020.8**	7178.5
Media		41757.9	**41324.6**	44313.7	43858.0	**42873.9**



(a) Problema estático.

(b) Problema dinámico.

Figura 5.9: Porcentaje de mejora de GRASP y MA respecto a PD.

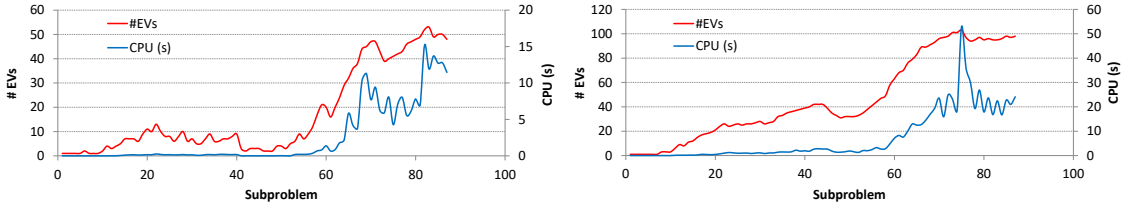
Respecto al tiempo de ejecución necesario para resolver el problema estático, el MA tarda 67 segundos en promedio, mientras que el algoritmo GA tarda 93 segundos, por lo que podemos concluir que el MA necesita un promedio de 28 % menos de tiempo de ejecución que el GA.

Tabla 5.10: Comparación con el estado del arte en las instancias del escenario 3. Cada valor representa la suma del tardiness (en horas) de las 30 instancias de cada grupo.

N	Δ	Problema Estático		Problema Dinámico		
		GRASP	MA	PD [38]	GRASP	MA
Instancias *Tipo* 1						
20	0.2	**18918.6**	19209.6	20704.7	20537.6	**20242.4**
	0.4	16737.5	**16628.5**	18001.7	18083.8	**17916.8**
	0.6	16293.4	**16218.8**	**17528.2**	17613.2	17573.4
	0.8	16189.4	**16136.2**	**17463.3**	17542.7	17524.5
30	0.2	**7764.2**	7791.9	9051.1	8752.9	**8569.1**
	0.4	6549.4	**6528.0**	7347.3	7290.4	**7283.5**
	0.6	6404.8	**6379.6**	7150.4	**7080.3**	7115.3
	0.8	6399.8	**6371.6**	7144.5	**7071.4**	7106.9
40	0.2	2504.9	**2422.5**	3478.1	3122.8	**3106.7**
	0.4	1862.7	**1853.4**	2373.1	**2318.3**	2328.6
	0.6	1825.1	**1810.3**	2276.7	**2225.8**	2247.6
	0.8	1826.9	**1810.2**	2276.7	**2224.1**	2246.7
Media		8606.4	**8596.7**	9566.3	9488.6	**9438.5**
Instancias *Tipo* 2						
20	0.2	131321.0	**129339.0**	138064.0	132960.7	**131037.0**
	0.4	57272.0	**56672.0**	62988.8	59611.2	**58225.8**
	0.6	37418.1	**37089.8**	42528.3	39366.3	**38574.6**
	0.8	31860.4	**31715.6**	33998.2	33931.6	**33360.3**
30	0.2	80438.3	**79182.4**	83169.5	82459.7	**80526.7**
	0.4	33313.1	**33084.2**	34799.7	34680.0	**34045.1**
	0.6	19697.9	**19592.6**	21321.1	20985.8	**20645.6**
	0.8	15852.8	**15812.5**	**16972.0**	17150.0	16979.7
40	0.2	56339.3	**55832.1**	58306.3	58082.5	**56737.6**
	0.4	22064.6	**21932.0**	22988.7	23091.7	**22710.2**
	0.6	11520.4	**11462.9**	12220.3	12292.9	**12168.1**
	0.8	8400.8	**8388.2**	**9127.3**	9265.2	9204.6
Media		42124.9	**41675.3**	44707.0	43656.5	**42851.3**

Al igual que en el GA, el tiempo de ejecución en el problema dinámico depende de los parámetros de la instancia y del número de vehículos del subproblema particular. El tiempo de ejecución promedio del MA es de 224 segundos por ejecución. Estos tiempos representan el tiempo total para resolver todos los subproblemas en un horizonte de 24 horas. En el GA el tiempo promedio es de 325 segundos por ejecución, y como hemos utilizado el mismo servidor para la ejecución de los experimentos, podemos concluir que el tiempo de ejecución del MA es aproximadamente un 31 % más bajo que el del GA.

Para saber si MA es apropiado para un entorno real, debemos verificar si el tiempo de ejecución necesario para resolver cualquier subproblema de la versión dinámica es inferior a los $\Delta T = 120s$ entre subproblemas consecutivos. Para ello hemos medido los tiempos en dos de los casos más difíciles: el primero es una instancia del escenario 1, *Tipo* 1, $N = 20$ y $\Delta = 0.2$ y el segundo es una instancia equivalente del *Tipo* 2, es decir, escenario 1, *Tipo* 2, $N = 20$ y $\Delta = 0.2$. En la Figura 5.10 mostramos los tiempos de ejecución necesarios para resolver cada subproblema (en



(a) Instancia de *Tipo* 1, $N = 20$ y $\Delta = 0.2$. (b) Instancia de *Tipo* 2, $N = 20$ y $\Delta = 0.2$.

Figura 5.10: Tiempo de ejecución y número de EVs en cada subproblema de una instancia dinámica.

azul) y el número de EVs planificados en cada uno (en rojo). En ambas figuras, el eje X muestra la secuencia de subproblemas, mientras que en el eje Y principal se indica el número de EVs y en el eje Y secundario el tiempo de ejecución en segundos. En el caso de la instancia de *Tipo* 1, que se muestra en la Figura 5.10(a), se resuelven 87 subproblemas, el número promedio de EVs en cada subproblema es 17, el tiempo promedio de ejecución es 2.41 segundos y el mayor tiempo necesario es 15.19 segundos en un subproblema con 53 EVs. En el caso de la instancia de *Tipo* 2, que se muestra en la Figura 5.10(b), se resuelven 90 subproblemas con un tamaño medio de 48 EVs, el tiempo promedio es de 7.98 segundos y el mayor es de 52.84 segundos en un subproblema con 103 EVs. Como estos tiempos aún están lejos de los 120 segundos que hay entre dos ejecuciones del planificador, podemos concluir que MA podría ser apropiado en un entorno real.

Finalmente, destacamos que los tiempos de ejecución de GRASP son ligeramente mayores que los de MA tanto en los problemas estáticos como dinámicos, y además sus resultados son generalmente peores, lo que confirma que MA es una metaheurística más eficaz.

Resultados del hABC

En [27] hemos realizado un estudio experimental para evaluar el algoritmo hABC y compararlo con los resultados publicados en el estado del arte. Previamente, en [25] habíamos publicado algunos resultados preliminares de una versión más básica de un algoritmo ABC sin hibridación.

Para buscar la mejor configuración de parámetros de entrada del algoritmo, de nuevo hemos realizado algunos experimentos utilizando un conjunto de 24 instancias, concretamente la primera instancia de cada grupo del escenario 1. Para estos experimentos, hemos ajustado la condición de parada para que el tiempo de cálculo de las diferentes configuraciones sea similar y comparable a la de otras metaheurísticas propuestas. La Tabla 5.11 resume los valores probados para cada parámetro y los mejores valores para las versiones estática y dinámica del problema.

Con la configuración de parámetros propuesta en las columnas "Mejor estático" y "Mejor dinámico" de la Tabla 5.11, y considerando como condición de parada 25 ciclos consecutivos sin mejorar la mejor solución obtenida hasta el momento vemos que el patrón de convergencia es apropiado. Como ejemplo, la Figura 5.11 muestra la evolución del tardiness total en una ejecución de una instancia del escenario 1, *Tipo* 1, $N = 20$ y $\Delta = 0.2$, considerando la versión estática del problema.

Para hacer una comparación de los resultados de hABC, utilizamos como referencias principales el algoritmo PD propuesto en [38], el MA propuesto en [24] (y

Tabla 5.11: Resumen de los valores probados para ajustar los parámetros del algoritmo hABC.

Parámetro	Valores Probados	Mejor Estático	Mejor Dinámico
Tamaño del torneo (t_{size})	5, 10, 15	10	15
Núm. de fuentes de alimento (SN)	100, 300, 500	300	100
Tipo de employed bee phase	e_meth1, e_meth2	e_meth2	e_meth2
Operador de cruce	SBX, PMX, Aleatorio	Aleatorio	Aleatorio
Máx. intentos mejora ($limit$)	25, 50, 75, 100	50	25
$o\_meth1 :: step\_size$	1, 2, 5, 10	5	2
$o\_meth1 :: max\_improv$	1, 2, 4, 8	2	4
$o\_meth2 :: step\_size$	2, 5, 10	5	2
$o\_meth2 :: max\_steps$	5, 10, 20	20	10
$o\_meth2 :: max\_improv$	2, 4, 8	4	4
Tipo de onlooker bee phase	o_meth1, o_meth2	o_meth1	o_meth1
max_step_perc búsqueda local	0.05, 0.1, 0.25	0.1	0.1
Búsqueda local en el paso final	Aplicar, No aplicar	Aplicar	Aplicar

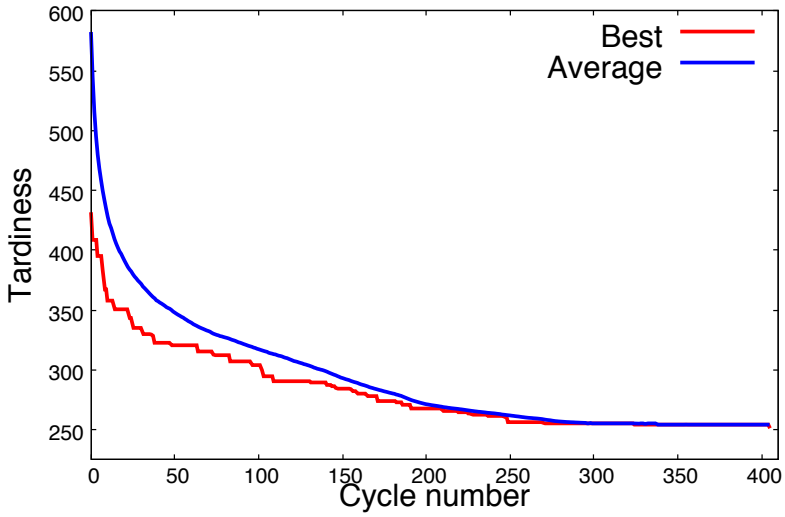


Figura 5.11: Evolución del mejor tardiness y del tardiness medio de la población de una ejecución del algoritmo hABC.

descrito en la Sección 4.2.4) y el ABC propuesto en [25]. Estos últimos algoritmos mejoran la calidad de las soluciones del PD a costa de incrementar el tiempo de ejecución. Utilizando la mejor configuración mostrada en la Tabla 5.11, hemos realizado un estudio experimental con todas las instancias del banco de problemas. Las Tablas 5.12, 5.13 y 5.14 muestran los resultados en las instancias de los escenarios 1, 2 y 3, respectivamente. Para el escenario 1, mostramos los resultados obtenidos por los algoritmos MA [24], ABC [25] y hABC. Para el algoritmo PD [38] recordemos que sus autores no abordan la versión estática del problema. Para los escenarios 2 y 3 mostramos valores del tardiness de hABC junto con los publicados para PD [38] y MA [24] para el problema dinámico, puesto que en [25] no hay resultados publicados para el algoritmo ABC en estos escenarios. Cada valor en las tablas representa la suma del tardiness (en horas) de las 30 instancias de cada grupo y los mejores valores obtenidos para cada subconjunto de instancias están remarcados en negrita.

Los resultados mostrados en las Tablas 5.12, 5.13 y 5.14 indican que nuestro algoritmo hABC supera a ABC [25] y PD [38] en todos los grupos de 30 instancias,

Tabla 5.12: Comparación de hABC con los métodos del estado del arte en las instancias del escenario 1.

N	Δ	Problema Estático			Problema Dinámico			
		MA [24]	ABC [25]	hABC	PD [38]	MA [24]	ABC [25]	hABC
Instancias *Tipo* 1								
20	0.2	5210.8	5331.7	**5089.0**	8386.3	6981.2	7027.9	**6948.1**
	0.4	**2509.2**	2586.1	2529.7	4120.4	3884.2	3898.1	**3877.4**
	0.6	2252.7	2258.2	**2250.6**	3670.6	3556.7	3558.0	**3547.8**
	0.8	2199.0	2200.4	**2194.1**	3590.9	3503.3	3509.1	**3494.5**
30	0.2	729.5	771.9	**712.6**	1959.3	**1444.8**	1445.7	1445.0
	0.4	**52.5**	76.8	75.4	421.2	367.6	366.6	**364.8**
	0.6	**34.3**	34.9	34.4	347.9	**316.2**	317.7	316.8
	0.8	**33.8**	34.1	33.7	347.6	**314.7**	316.4	315.9
40	0.2	**216.9**	238.6	221.4	735.0	541.6	545.2	**540.3**
	0.4	**0.0**	**0.0**	**0.0**	14.0	7.7	7.8	**7.4**
	0.6	**0.0**	**0.0**	**0.0**	**3.4**	**3.4**	**3.4**	**3.4**
	0.8	**0.0**	**0.0**	**0.0**	**3.4**	**3.4**	**3.4**	**3.4**
Media		1103.2	1127.7	**1095.1**	1966.7	1744.0	1749.9	**1738.7**
Instancias *Tipo* 2								
20	0.2	**122,615.0**	123,409.0	122,690.0	128,185.0	124,075.0	124,168.0	**123,934.0**
	0.4	**43991.1**	44,152.7	44,030.0	46,319.3	45,127.2	45,183.9	**45,104.0**
	0.6	**20,526.9**	20,597.0	20,545.1	22,966.8	21,822.6	21,847.7	**21,788.5**
	0.8	**12,762.8**	12,734.1	12,727.2	14,573.1	14,209.9	14,212.1	**14,176.8**
30	0.2	**69,610.6**	69,954.3	69,665.3	72,860.8	70,894.5	70,942.6	**70,853.6**
	0.4	**20,031.7**	20,097.6	20,043.6	21,479.9	21,130.3	21,150.5	**21,109.1**
	0.6	7030.5	7033.0	**7019.0**	8088.9	7921.5	7923.0	**7905.2**
	0.8	**3527.9**	3536.4	3530.8	4486.3	4389.3	4391.0	**4384.4**
40	0.2	**43,981.8**	44,146.7	44,024.5	46,135.4	45,139.2	45,192.6	**45,113.7**
	0.4	9760.7	9775.7	**9753.1**	10,869.3	10,654.6	10,669.2	**10,635.7**
	0.6	2851.1	2855.7	**2850.4**	3599.1	3515.6	3515.1	**3514.8**
	0.8	872.3	876.3	**872.0**	1635.5	1572.1	1574.9	**1571.7**
Media		**29796.9**	29930.7	29812.6	31766.6	30871.0	30897.5	**30841.0**

tanto en la versión estática del problema como en la dinámica. Nuestro algoritmo también supera a MA [24] en la mayoría de las instancias, aunque las diferencias en el problema estático considerando el escenario 1 son pequeñas.

El tiempo medio de ejecución de hABC en la versión estática es de 28.5 segundos para las instancias de *Tipo* 1, mientras que para las de *Tipo* 2 es de 100.8 segundos. Estos tiempos de ejecución son superiores a los del MA, que necesita 21.7 segundos en promedio para las instancias de *Tipo* 1 y 77.5 segundos para las de *Tipo* 2.

Respecto a los tiempos de ejecución de la versión dinámica, de nuevo hemos comprobado que las instancias que requieren un mayor tiempo son las de *Tipo* 2, $N = 20$ y $\Delta = 0.2$. Para verificar que hABC se puede ejecutar en un entorno real medimos el tiempo medio que necesita para resolver una instancia con esos parámetros, que es de 860.1 segundos; el tiempo medio de ejecución de cada subproblema es de 1.88 segundos, siendo 12.34 segundos en el peor de los casos. Claramente, estos tiempos son mucho más bajos que dos minutos y, por lo tanto, podemos concluir que hABC se puede utilizar en un entorno real. Los otros métodos metaheurísticos requieren tiempos de ejecución similares: MA necesitó un promedio de 1.23 segundos por

Tabla 5.13: Comparación de hABC con los métodos del estado del arte en las instancias del escenario 2.

N	Δ	Problema Estático		Problema Dinámico		
		MA [24]	hABC	PD [38]	MA [24]	hABC
Instancias Tipo 1						
20	0.2	14,411.5	**14,097.9**	16,886.0	16,176.3	**15,933.8**
	0.4	**11,703.6**	11,713.9	14,131.2	13,915.9	**13,869.9**
	0.6	**11,328.3**	11,335.7	13,648.3	13,559.1	**13,534.8**
	0.8	**11,242.1**	11,242.9	13,547.9	13,492.7	**13,464.7**
30	0.2	3920.6	**3887.3**	6394.7	5821.5	**5798.8**
	0.4	2726.2	**2717.1**	4824.3	4688.4	**4687.2**
	0.6	2592.8	**2588.3**	4668.6	4560.3	**4559.3**
	0.8	2591.4	**2587.2**	4672.6	4559.0	**4556.7**
40	0.2	**638.0**	662.1	1951.4	1593.5	**1590.0**
	0.4	**186.4**	**186.4**	1212.7	1102.0	**1101.5**
	0.6	172.2	**171.5**	1210.7	1103.3	**1102.8**
	0.8	171.9	**171.6**	1210.6	1102.7	**1102.5**
Media		5140.4	**5113.5**	7029.9	6806.2	**6775.2**
Instancias Tipo 2						
20	0.2	**137,204.0**	137,271.0	143,883.0	139,192.0	**138,961.0**
	0.4	57,453.4	**57,295.2**	62,246.2	59,194.0	**59,065.2**
	0.6	34,548.2	**34,499.5**	39,869.4	36,298.4	**36,231.7**
	0.8	27,735.1	**27,711.0**	30,887.2	29,791.0	**29,747.5**
30	0.2	82,970.0	**83,224.2**	86,392.0	84,470.0	**84,297.4**
	0.4	32,714.9	**32,659.2**	34,475.4	33,936.8	**33,891.5**
	0.6	17,077.3	**17,069.9**	19,355.7	18,709.8	**18,682.5**
	0.8	12,364.5	**12,359.5**	14,375.1	14,201.7	**14,182.4**
40	0.2	57,652.0	**57,630.0**	59,775.0	58,714.1	**58,591.2**
	0.4	20,786.9	**20,765.5**	22,254.0	21,984.2	**21,947.1**
	0.6	9582.4	**9581.7**	10,991.3	10,815.9	**10,808.5**
	0.8	**5806.0**	5814.1	7260.5	7178.5	**7170.9**
Media		41324.6	**41323.4**	44313.7	42873.9	**42798.1**

subproblema y 10.26 segundos en el peor de los casos, y ABC necesitó 1.97 segundos en promedio y 15.03 segundos en el peor de los casos. Sin embargo, recordemos que PD es el algoritmo más rápido, con menos de 0.012 segundos en todas las ejecuciones, aunque también obtuvo los peores resultados.

5.2. El problema con incertidumbre

5.2.1. Banco de instancias

Para el problema con tiempos de carga difusos hemos tomado como instancias de referencia versiones difusas de las instancias del problema determinista, simulando incertidumbre en los tiempos de carga mediante el método de fuzzificación propuesto por Ghrayeb en [28]. Cada tiempo de carga difuso se genera de tal forma que el valor modal a_2 del TFN es el tiempo de carga original, mientras que el soporte $[a_1, a_3]$ son dos valores reales calculados de la siguiente forma: a_1 es un valor aleatorio en el intervalo $[0.85 \times a_2, a_2]$ y a_3 es un valor aleatorio en el intervalo $[a_2, 1.25 \times a_2]$.

Tabla 5.14: Comparación del hABC con los métodos del estado del arte en las instancias del escenario 3.

N	Δ	Problema Estático		Problema Dinámico		
		MA [24]	hABC	PD [38]	MA [24]	hABC
Instancias Tipo 1						
20	0.2	19,209.6	**18,874.0**	20,704.7	20,242.4	**20,055.1**
	0.4	16,628.5	**16,632.1**	18,001.7	17,916.8	**17,875.2**
	0.6	16,218.8	**16,227.3**	**17,528.2**	17,573.4	17,553.6
	0.8	16,136.2	**16,134.3**	**17,463.3**	17,524.5	17,505.9
30	0.2	7791.9	**7733.0**	9051.1	8569.1	**8556.5**
	0.4	6528.0	**6524.1**	7347.3	7283.5	**7276.2**
	0.6	6379.6	**6380.1**	7150.4	7115.3	**7113.5**
	0.8	6371.6	**6371.2**	7144.5	7106.9	**7103.8**
40	0.2	2422.5	**2438.5**	3478.1	**3106.7**	3108.5
	0.4	1853.4	**1853.4**	2373.1	**2328.6**	2329.6
	0.6	1810.3	**1810.0**	2276.7	2247.6	**2247.2**
	0.8	1810.2	**1810.0**	2276.7	2246.7	**2246.4**
Media		8596.7	**8565.6**	9566.3	9438.5	**9414.3**
Instancias Tipo 2						
20	0.2	129,339.0	**129,171.0**	138,064.0	131,037.0	**130,903.0**
	0.4	56,672.0	**56,513.3**	62,988.8	58,225.8	**58,080.9**
	0.6	37,089.8	**37,034.4**	42,528.3	38,574.6	**38,507.2**
	0.8	31,715.6	**31,705.2**	33,998.2	33,360.3	**33,313.8**
30	0.2	**79,182.4**	79,359.1	83,169.5	80,526.7	**80,393.4**
	0.4	33,084.2	**33,021.5**	34,799.7	34,045.1	**33,990.5**
	0.6	19,592.6	**19,559.8**	21,321.1	20,645.1	**20,622.7**
	0.8	15,812.5	**15,805.1**	16,972.0	16,979.7	**16,962.8**
40	0.2	55,832.1	**55,747.9**	58,306.3	56,737.6	**56,635.1**
	0.4	21,932.0	**21,901.8**	22,988.7	22,710.2	**22,680.2**
	0.6	11,462.9	**11,455.3**	12,220.3	12,168.1	**12,161.5**
	0.8	8388.2	**8387.9**	**9127.3**	9204.6	9198.8
Media		41675.3	**41638.5**	44707.0	42851.3	**42787.5**

5.2.2. Resultados del fGA

En [27] hemos realizado un estudio experimental para evaluar el algoritmo fGA, descrito en la Sección 4.2.6. Después de un estudio paramétrico inicial, hemos fijado los parámetros del algoritmo de la siguiente forma: el tamaño de la población es de 200 individuos, establecemos la condición de parada en 25 generaciones consecutivas sin mejorar el mejor individuo encontrado hasta ese momento $max_{iter} = 25$ o un individuo con tardiness $\widehat{0}$. Las probabilidades de cruce y mutación son $P_c = 0.8$ y $P_m = 0.1$ respectivamente, y el tamaño del torneo para generar la población inicial heurística se establece en $tSize = 8$.

A continuación analizamos la evolución del fGA, comparando sus resultados con un algoritmo heurístico sin evolución que genera tantos grupos de 200 individuos como generaciones necesita el fGA para converger. Cada grupo está formado por individuos aleatorios e individuos heurísticos generados con las versiones estocásticas de las reglas de prioridad DDR y EST en la misma proporción que en la población

inicial del fGA. Para hacer las comparaciones de la forma más justa posible, este algoritmo heurístico aplica elitismo al mantener siempre la mejor solución encontrada hasta el momento.

La Figura 5.12 ilustra la convergencia del fGA para una instancia del escenario 1, *Tipo* 1, $N = 20$ y $\Delta = 0.2$. El fGA y el algoritmo heurístico presentan un patrón de evolución similar en todas las instancias. El gráfico corresponde a la evolución del fitness medio de la población (es decir, el promedio de la esperanza del tardiness total $E[\widehat{T}]$) de 30 ejecuciones tanto del fGA como del algoritmo heurístico. Está claro que la mejora con respecto a la mejor solución inicial (la misma para ambos métodos) es mayor con el fGA (45.15 % de mejora) que con el algoritmo heurístico (12.86 %).

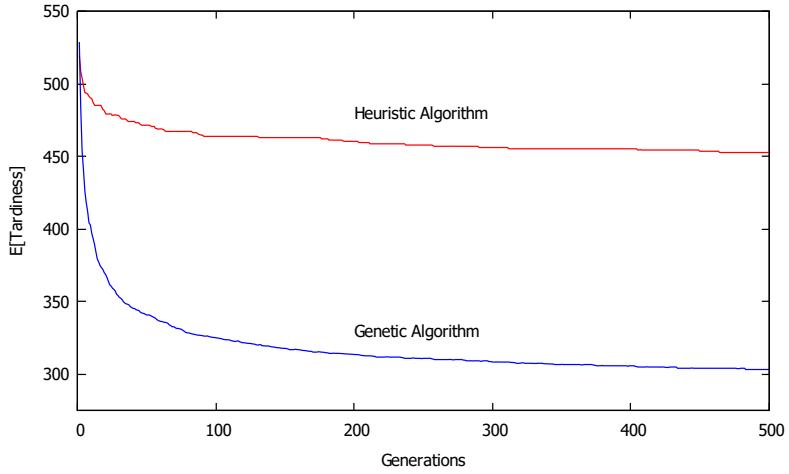


Figura 5.12: Evolución promedio del fitness para el fGA (en azul) y el algoritmo heurístico (en rojo).

Debido a que una de las aportaciones de [27] es la formulación del problema de planificación de vehículos con tiempos de carga difusos, no existen otros algoritmos en la literatura con los que comparar los resultados obtenidos. Por lo tanto, para evaluar el rendimiento del fGA comparamos sus resultados con los obtenidos por el algoritmo heurístico descrito anteriormente, que esencialmente consiste en generar individuos aleatorios e individuos heurísticos. En las Tablas 5.15 y 5.16 mostramos los resultados obtenidos por ambos algoritmos en condiciones equivalentes, es decir, igual número de individuos evaluados. La Tabla 5.15 indica el tardiness difuso $\widehat{T}$ junto con el valor esperado $E[\widehat{T}]$ de la mejor solución encontrada para cada algoritmo (columnas 3, 4 para el fGA y 5, 6 para el método heurístico). La última columna corresponde a la mejora (en porcentaje) del fGA con respecto al algoritmo heurístico. Podemos ver que en las instancias de *Tipo* 1 esta mejora oscila entre 6.2 % y un poco más de 69 %, con una mejora promedio cercana al 33 %. En las instancias de *Tipo* 2 la mejora es más constante, con un valor medio del 21.42 %. La Tabla 5.16 contiene la esperanza del tardiness medio de las mejores soluciones del fGA y del método heurístico, junto con la desviación estándar y el tiempo medio de CPU (en segundos). La última columna indica el porcentaje de mejora medio del fGA con respecto al algoritmo heurístico. Podemos ver que, con respecto a la esperanza del tardiness medio, el fGA es en promedio un 40.52 % mejor que el algoritmo heurístico en las instancias de *Tipo* 1 y un 21.38 % mejor en las instancias de *Tipo* 2.

Tabla 5.15: Mejores valores del tardiness total para el fGA y el algoritmo heurístico.

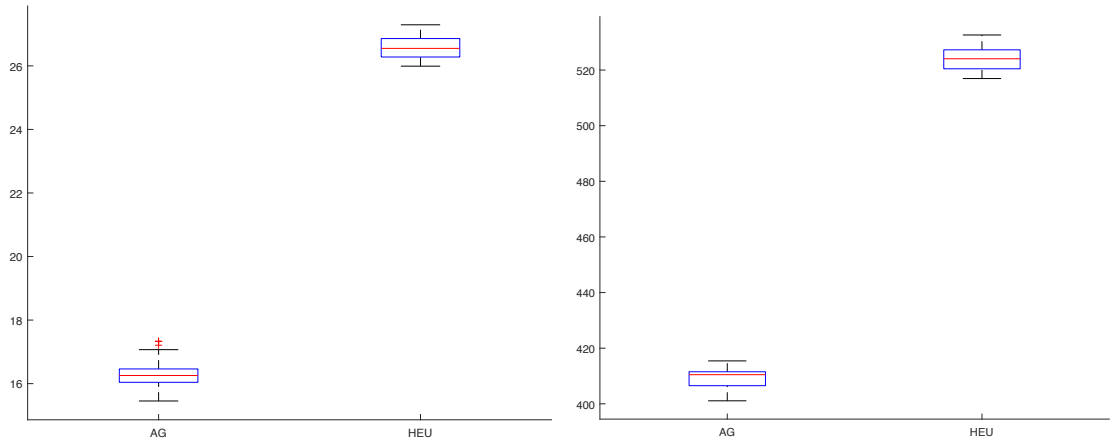
N	Δ	fGA		Heurístico		% Mejora
		Mejor $\widehat{T}$	$E[\widehat{T}]$	Mejor $\widehat{T}$	$E[\widehat{T}]$	
Instancias *Tipo* 1						
20	0.2	(197.7, 278.3, 430.9)	296.3	(335.9, 407.3, 545.7)	424.1	30.12
	0.4	(77.35, 126.3, 244.8)	143.7	(114.1, 165.2, 282.8)	181.8	20.97
	0.6	(62.65, 109.9, 219.8)	125.5	(72.13, 118.6, 227.1)	134.1	6.38
	0.8	(61.41, 109.6, 219.7)	125.1	(72.14, 118.6, 223.9)	133.3	6.19
30	0.2	(28.89, 49.37, 102.7)	57.61	(81.29, 110.3, 177.6)	119.9	51.96
	0.4	(1.300, 8.333, 43.84)	15.45	(8.342, 17.43, 60.75)	25.99	40.54
	0.6	(0.250, 6.616, 42.18)	13.91	(5.483, 12.96, 54.18)	21.39	34.97
	0.8	(0.716, 6.633, 41.64)	13.90	(5.483, 12.96, 54.18)	21.39	35.01
40	0.2	(3.013, 8.470, 27.19)	11.78	(22.83, 34.46, 60.54)	38.07	69.05
	0.4	(0, 0, 0)	0.00	(0, 0, 0)	0.00	-
	0.6	(0, 0, 0)	0.00	(0, 0, 0)	0.00	-
	0.8	(0, 0, 0)	0.00	(0, 0, 0)	0.00	-
Instancias *Tipo* 2						
20	0.2	(4481, 4644, 4896)	4666	(5350, 5514, 5757)	5534	15.67
	0.4	(1560, 1707, 1949)	1731	(1961, 2117, 2350)	2137	19.00
	0.6	(701.9, 836.1, 1056)	857.7	(935.2, 1037, 1210)	1054	18.70
	0.8	(372.4, 489.9, 691.1)	510.8	(487.0, 592.4, 778.7)	612.6	16.61
30	0.2	(2473, 2667, 2967)	2694	(3019, 3180, 3418)	3199	15.81
	0.4	(681.2, 804.5, 1000)	822.7	(909.9, 1007, 1174)	1024	19.71
	0.6	(177.5, 252.2, 394.9)	269.2	(270.4, 341.1, 478.9)	357.9	24.78
	0.8	(43.95, 81.29, 182.4)	97.24	(68.12, 114.8, 216.8)	128.6	24.43
40	0.2	(1564, 1705, 1931)	1726	(1961, 2117, 2350)	2137	19.20
	0.4	(302.6, 385.0, 531.6)	401.0	(412.5, 501.3, 652.6)	516.9	22.41
	0.6	(28.84, 64.54, 153.3)	77.81	(58.15, 100.2, 187.7)	111.6	30.29
	0.8	(0.299, 6.307, 57.50)	17.60	(3.506, 15.04, 67.64)	25.30	30.43

Hemos realizado un análisis más detallado con una instancia de cada tipo, concretamente, una instancia de *Tipo* 1, $N = 30$ y $\Delta = 0.4$ y otra de *Tipo* 2, $N = 40$ y $\Delta = 0.4$. Para cada caso, hemos obtenido el valor esperado del tardiness de 30 ejecuciones de cada algoritmo. Después de verificar la normalidad, hemos realizado un t-test para comprobar que en ambos casos fGA obtiene resultados significativamente mejores que el algoritmo heurístico. En la Figura 5.13 podemos ver los diagramas boxplot correspondientes a ambas instancias.

Hay que destacar que los valores del tardiness en las Tablas 5.15 y 5.16 reflejan que la dificultad de una instancia depende de la carga de EVs en cada línea L_i y de los valores de los parámetros N y Δ. Como se esperaba, es más fácil encontrar buenas soluciones para las instancias de *Tipo* 1, donde la carga en todas las líneas es más uniforme que en las instancias de *Tipo* 2. También está claro que las instancias que tienen un número máximo de puntos de carga activos por línea N más bajo y una restricción de desequilibrio más estricta, representada por valores bajos de Δ, son más difíciles de resolver. De hecho, podemos encontrar dos instancias patológicas, ambas con cargas muy desiguales, una pequeña proporción de puntos de carga activos por línea y una restricción de equilibrio de carga muy estricta. Estas son instancias de *Tipo* 2, donde la primera línea, L_1, soporta el 60 % de la demanda, con 108 EVs, el 30 % (54 EVs) de la demanda recae sobre la segunda línea, L_2 y

Tabla 5.16: Media de la esperanza del tardiness del fGA y del algoritmo heurístico.

N	Δ	fGA		Heurístico		% Mejora
		Media±SD	CPU(s)	Media±SD	CPU(s)	
Instancias *Tipo* 1						
20	0.2	336.0±12.17	40	458.5±7.17	194	29.91
	0.4	149.4±1.39	50	192.3±2.30	199	21.50
	0.6	129.3±0.91	51	137.6±0.66	201	7.00
	0.8	127.1±0.46	55	134.9±0.40	201	6.19
30	0.2	77.74±4.74	54	144.8±5.13	199	52.61
	0.4	17.33±0.46	35	27.29±0.36	206	38.51
	0.6	17.21±0.71	29	23.08±0.36	206	34.04
	0.8	16.66±0.60	30	23.08±0.36	206	34.73
40	0.2	19.41±2.09	44	45.26±2.18	202	64.00
	0.4	0.091±0.03	9	0.137±0.03	206	69.03
	0.6	0.091±0.04	8	0.133±0.03	206	64.33
	0.8	0.091±0.04	8	0.133±0.03	206	64.33
Instancias *Tipo* 2						
20	0.2	4777±24.00	186	5658±30.41	194	15.60
	0.4	1801±14.58	145	2203±15.55	198	19.10
	0.6	896.7±8.47	111	1085±6.93	200	18.66
	0.8	528.5±4.39	88	631.5±4.78	201	16.56
30	0.2	2772±20.60	173	3343±29.70	196	16.95
	0.4	859.9±9.24	122	1055±6.43	202	19.44
	0.6	279.6±2.53	54	370.8±3.30	207	24.85
	0.8	104.9±1.97	40	136.5±1.87	208	23.50
40	0.2	1794±15.12	148	2186±12.29	198	18.54
	0.4	415.4±3.69	92	532.6±4.17	205	21.87
	0.6	83.86±1.57	42	118.3±1.65	210	30.63
	0.8	20.74±0.69	40	28.19±0.62	210	30.80



(a) Instancia de *Tipo* 1, $N = 30$ y $\Delta = 0.4$. (b) Instancia de *Tipo* 2, $N = 40$ y $\Delta = 0.4$.

Figura 5.13: Boxplot de los resultados de 30 ejecuciones del fGA y del algoritmo heurístico en dos instancias representativas.

solo el 10 % (18 EVs) recae sobre el tercera línea, L_3. Además, en estas instancias el número máximo de puntos de carga activos es muy bajo, con valores $N = 20$ y $N = 30$ (respectivamente, 33 % y 50 % de los 60 puntos disponibles en cada línea),

y una restricción de desbalanceo muy estricta con $\Delta = 0.2$. Como consecuencia, una vez que los EVs que van a la tercera línea L_3 (solo 18) terminan de cargarse, el número de EVs que se puede cargar en cualquier momento en cualquiera de las otras dos líneas se limita a solo 4 o 6 (con $N = 20$ y $N = 30$ respectivamente). Esto crea un cuello de botella y los retrasos posteriores en L_1 y L_2, lo que es especialmente problemático para L_1, donde se debe cargar un total de 90 EVs. Como se mencionó anteriormente, estos casos son patológicos y no se puede esperar que se resuelvan satisfactoriamente. Sin embargo, plantean casos extremos que sirven como desafío para cualquier método de resolución que pueda proponerse para el problema en cuestión.

Finalmente, para evaluar la dificultad de encontrar una solución a este problema con métodos exactos, en [27] proporcionamos una formulación alternativa de programación lineal entera (ILP) que implementamos y ejecutamos utilizando el software IBM ILOG CPLEX Optimizer [42]. Como podemos ver en la Tabla 5.15, las instancias más fáciles son las de *Tipo* 1 con los valores más elevados para N y Δ. De hecho, tanto el fGA como el algoritmo heurístico encuentran una solución óptima para las instancias de *Tipo* 1 con $N = 40$ y $\Delta = 0.8$. Hemos generado nuevas instancias más pequeñas de *Tipo* 1 al "podar" las existentes, considerando solo los primeros 20 EVs de los 180 originales, dejando el parámetro de desbalanceo $\Delta = 0.8$ sin cambios y redimensionando el número máximo de vehículos a $N = 5$, que es proporcionalmente menos estricto que $N = 40$ para 180 EVs. Para estas instancias muy simples, IBM ILOG CPLEX Optimizer tarda 12 horas en proporcionar una solución. Para instancias un poco más grandes con 30 EVs, $N = 7$ y $\Delta = 0.8$ la misma formulación no puede proporcionar una solución factible después de estar ejecutándose durante 12 horas. Estos ultimos experimentos motivan el enfoque seguido a lo largo de esta tesis: la utilización de técnicas metaheurísticas para obtener buenas soluciones en un tiempo razonable.

Capítulo 6

Conclusiones

6.1. Aportaciones de la tesis

A lo largo de esta tesis hemos abordado el problema de planificación de la carga de EVs presentado en [38], y que está inspirado en un entorno real en el que varios EVs pueden requerir carga de un sistema eléctrico instalado en un garaje. Este problema consiste en asignar instantes de inicio a cada uno de los vehículos, con el fin de minimizar el retraso total respecto a la hora de recogida introducida por el usuario. El entorno tiene una serie de restricciones, como por ejemplo un número máximo de vehículos que se pueden cargar simultáneamente y restricciones de desequilibrio entre las tres líneas de la fuente de alimentación eléctrica. Hemos considerado dos variantes del problema: la estática, en la que sabemos de antemano el tiempo de llegada, el tiempo de carga y el instante de recogida de cada vehículo, y la dinámica, donde no tenemos estos datos. Aunque la variante estática no es realista, su estudio es relevante ya que puede proporcionar información útil para saber cuánto se puede mejorar la variante dinámica. También hemos planteado un primer enfoque para incorporar incertidumbre al problema, con el objetivo de reducir la brecha entre el modelo académico y el entorno real. En particular, hemos considerado tiempos de carga inciertos modelados como números difusos triangulares. Esto ha dado como resultado una nueva formulación del problema, basada en la versión determinista, pero con algunos cambios importantes, por ejemplo, en la traducción de las restricciones.

Hemos comenzado proponiendo un algoritmo genético para la versión determinista, con algunos operadores diseñados específicamente para tratar nuestro problema de manera eficiente. Hemos analizado nuestra propuesta y la hemos comparado con el único algoritmo existente hasta el momento para este problema, que es el enfoque de descomposición propuesto en [38], obteniendo resultados significativamente mejores, aunque utilizando un mayor tiempo computacional. El constructor de planificaciones, los operadores de cruce y mutación, y la forma de generar una población inicial diversa son contribuciones de esta tesis.

A continuación hemos propuesto dos métodos que utilizan búsqueda local: uno basado en una metaheurística GRASP que consiste en una fase constructiva seguida de una fase de mejora, y un algoritmo memético. Hemos definido cuatro vecindades diferentes para la búsqueda local, y hemos considerado tanto una estrategia de escalada como una búsqueda en entorno variable VND. En el estudio empírico, hemos visto que la estrategia evolutiva (algoritmo memético) produce mejores resultados que la estrategia de inicio múltiple (algoritmo GRASP), lo que sugiere que hacer

evolucionar una población de soluciones durante una serie de generaciones es mejor que simplemente mejorar un conjunto de soluciones aleatorias o heurísticas. También podemos concluir que las metaheurísticas híbridas son métodos eficientes, ya que la búsqueda local ha sido capaz de mejorar los resultados finales obtenidos por el algoritmo genético. En cuanto a la búsqueda local, los resultados obtenidos por la metaheurística Variable Neighborhood Descent (VND) son mejores que los de una escalada simple, siempre que seleccionemos adecuadamente las estructuras de vecindad. Además, hemos visto que los beneficios de utilizar un planificador mejorado con un refinamiento local son relevantes en una estrategia de inicio múltiple como GRASP, o en un algoritmo genético simple, pero no tanto en metaheurísticas avanzadas combinadas con búsqueda local. Creemos que las principales razones de la eficiencia de nuestros enfoques, en particular el algoritmo memético, son el equilibrio adecuado entre la diversificación y la intensificación en la búsqueda. La diversificación se logra mediante la utilización de varios métodos para generar soluciones iniciales buenas y diversas, el operador de cruce y la estrategia de reemplazo, mientras que la intensificación es provista por una combinación de estructuras de vecindad eficientes utilizadas en la búsqueda VND.

Siguiendo con la línea de las metaheurísticas híbridas, hemos propuesto una basada en el comportamiento de las abejas, que toma prestadas algunas técnicas de selección y cruce de los algoritmos genéticos, y la hemos mejorado añadiéndole una búsqueda local. Mediante un estudio experimental, hemos analizado la propuesta y comprobado que esta metaheurística obtiene resultados excelentes, y en muchos casos supera al resto de métodos propuestos en esta tesis. En este caso la intensificación se logra mediante la fase de abeja observadora y la búsqueda local, mientras que la diversificación se consigue en la generación inicial de soluciones y en la fase de abeja exploradora. Por otra parte, el método propuesto para elegir qué fuentes de alimento se combinan en la fase de abeja trabajadora parece lograr un buen equilibrio entre intensificación y diversificación, propiciando elegir buenas soluciones pero teniendo en cuenta su diversidad.

Finalmente, hemos incorporado incertidumbre al problema considerando tiempos de carga inciertos modelados como números difusos triangulares, y hemos redefinido el problema. Para resolverlo, hemos propuesto un algoritmo genético basado en el algoritmo para la versión determinista. Aunque el esquema general sigue siendo el mismo, se ha propuesto un nuevo generador de planificaciones especialmente diseñado para la versión difusa. Hemos presentado un estudio experimental sobre nuevas instancias inspiradas en datos reales que muestran la convergencia correcta del algoritmo genético, así como su buen rendimiento.

Es destacable que el tiempo computacional requerido por todos los métodos presentados en esta tesis es razonable, y por lo tanto son apropiados para su utilización en entornos reales.

6.2. Trabajo futuro

A lo largo del estudio experimental, hemos visto que el tardiness obtenido al resolver la versión estática del problema es mucho menor que el de la versión dinámica. Esto nos lleva a pensar que puede haber mejores formas de abordar el problema dinámico. Tal vez si pudiéramos anticipar algunas llegadas de vehículos, podríamos evitar algunos cuellos de botella y, por lo tanto, planificar los vehícu-

los de una manera más eficiente. Otras opciones para futuras investigaciones son considerar características adicionales del problema real. Como por ejemplo, que los usuarios puedan recoger el vehículo antes del instante de recogida, o que la batería pueda cargarse completamente antes del tiempo de carga esperado. También sería interesante considerar que la potencia contratada pueda cambiar con el tiempo o que los vehículos se puedan cargar a una tasa no constante. Además, planeamos considerar nuevas funciones objetivo, como minimizar el desequilibrio entre las líneas o el consumo máximo, además de minimizar el tardiness total. Por lo tanto, se requerirá modelar y resolver el problema en el marco de los problemas de optimización multiobjetivo.

Por otra parte, la redefinición del problema de carga con incertidumbre constituye un primer enfoque que abre muchas líneas para futuras investigaciones. En primer lugar, nos gustaría establecer vínculos entre las restricciones del problema y la teoría de la posibilidad, lo cual podría sugerir nuevos y mejores algoritmos de creación de planificaciones. También es posible definir formulaciones alternativas para algunas restricciones a las que se podría incorporar incertidumbre, en particular, para el equilibrio entre líneas. Con respecto al método de resolución, podría mejorarse combinando el algoritmo genético con una búsqueda local especialmente definida para el problema con incertidumbre, lo que requeriría definir buenas estructuras de vecindad y métodos eficientes de evaluación de los vecinos. Otra perspectiva interesante, como se indica en [5], sería considerar la robustez de las soluciones obtenidas, ya sea como una función objetivo que habría que optimizar, o como un criterio para comparar las soluciones difusas y las deterministas. En resumen, nuestro objetivo es incorporar nuevas características que acerquen el problema a la situación real.

Bibliografía

[1] S. Abdullah and M. Abdolrazzagh-Nezhad. Fuzzy job-shop scheduling problems: A review. *Information Sciences*, 278:380–407, 2014.

[2] M. Ansari, A. T. Al-Awami, E. Sortomme, and M. A. Abido. Coordinated bidding of ancillary services for vehicle-to-grid using fuzzy optimization. *IEEE Transactions on Smart Grid*, 6(1):261–270, 2015.

[3] S. Balin. Parallel machine scheduling with fuzzy processing times using a robust genetic algorithm and simulation. *Information Sciences*, 181:3551–3569, 2011.

[4] M. Brunelli and J. Mezei. How different are ranking methods for fuzzy numbers? A numerical study. *International Journal of Approximate Reasoning*, 54:627–639, 2013.

[5] R. Burdett and E. Kozan. Techniques to effectively buffer schedules in the face of uncertainties. *Computers & Industrial Engineering*, 87:16–29, 2015.

[6] S. Chanas and P. Zieliński. Critical path analysis in the network with fuzzy activity times. *Fuzzy Sets and Systems*, 122:195–204, 2001.

[7] D. Dallinger. *Plug-in Electric Vehicles Integrating Fluctuating Renewable Electricity*. Kassel University Press, 2013.

[8] D. Dallinger and M. Wietschel. Grid integration of intermittent renewable energy sources using price-responsive plug-in electric vehicles. *Renewable and Sustainable Energy Reviews*, 16(5):3370–3382, 2012.

[9] D. Dubois, H. Fargier, and P. Fortemps. Fuzzy scheduling: Modelling flexible constraints vs. coping with incomplete knowledge. *European Journal of Operational Research*, 147:231–252, 2003.

[10] D. Dubois, H. Fargier, and P. Fortemps. Scheduling under flexible constraints and uncertain data: the fuzzy approach. In *Production Scheduling*, chapter 11, pages 301–332. Wiley, 2008.

[11] D. Dubois and H. Prade. Weighted minimum and maximum operations in fuzzy set theory. *Information Sciences*, 39:205–210, 1986.

[12] D. Dubois and H. Prade. The mean value of a fuzzy number. *Fuzzy Sets and Systems*, 24:279–300, 1987.

[13] EDSO. Position paper on electric vehicles charging infrastructure. Technical report, European Distribution system Operators for Smart Grids (EDSO), 10 April 2012.

[14] A. T. Ernst, H. Jiang, M. Krishnamoorthy, and S. D. Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research*, 153:3–27, 2004.

[15] M. Esmaili and A. Goldoust. Multi-objective optimal charging of plug-in electric vehicles in unbalanced distribution networks. *International Journal of Electrical Power & Energy Systems*, 73:644–652, 2015.

[16] Z. Fan. A distributed demand response algorithm and its application to phev charging in smart grids. *IEEE Transactions on Smart Grid*, 3(3):1280–1290, 2012.

[17] P. Fortemps. Jobshop scheduling with imprecise durations: a fuzzy approach. *IEEE Transactions of Fuzzy Systems*, 7:557–569, 1997.

[18] J. Foster and M. Caramanis. Optimal power market participation of plug-in electric vehicles pooled by distribution feeder. *IEEE Transactions on Power Systems*, 28(3):2065–2076, 2013.

[19] L. Gan, U. Topcu, and S. Low. Optimal decentralized protocol for electric vehicle charging. In *Proceedings IEEE Conference on Decision and Control and European Control Conference CDC-ECE*, pages 5798–5804. IEEE, 2011.

[20] L. Gan, U. Topcu, and S. Low. Optimal decentralized protocol for electric vehicle charging. *IEEE Transactions on Power Systems*, 28(2):940–951, 2013.

[21] J. Gao, L. Sun, and M. Gen. A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. *Computers & Operations Research*, 35:2892–2907, 2008.

[22] J. García-Villalobos, I. Zamora, J. San Martin, F. Asensio, and V. Aperribay. Plug-in electric vehicles in electric distribution networks: A review of smart charging approaches. *Renewable and Sustainable Energy Reviews*, 38:717–731, 2014.

[23] J. García-Álvarez, M. A. González, and C. R. Vela. A genetic algorithm for scheduling electric vehicle charging. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2015)*, pages 393–400, 2015.

[24] J. García-Álvarez, M. A. González, and C. R. Vela. Metaheuristics for solving a real-world electric vehicle charging scheduling problem. *Applied Soft Computing*, 65:292–306, 2018.

[25] J. García-Álvarez, M. A. González, C. R. Vela, and R. Varela. *Electric Vehicle Charging Scheduling Using an Artificial Bee Colony Algorithm*, pages 115–124. Springer International Publishing, 2017.

[26] J. García-Álvarez, M. A. González, C. R. Vela, and R. Varela. Electric vehicle charging scheduling by an enhanced artificial bee colony algorithm. *Energies*, 11(10), 2018.

[27] J. García-Álvarez, I. González-Rodríguez, C. R. Vela, M. A. González, and S. Afsar. Genetic fuzzy schedules for charging electric vehicles. *Computers & Industrial Engineering*, 121:51 – 61, 2018.

[28] O. A. Ghrayeb. A bi-criteria optimization: minimizing the integral value and spread of the fuzzy makespan of job shop scheduling problems. *Applied Soft Computing*, 2(3):197–210, 2003.

[29] D. Goldberg. *Genetic algorithms in search. Optimization and machine learning.* Addison-Wesley, 1985.

[30] I. Grau Unda, P. Papadopoulos, S. Skarvelis-Kazakos, L. Cipcigan, N. Jenkins, and E. Zabala. Management of electric vehicle battery charging in distribution networks with multi-agent systems. *Electric Power Systems Research*, 110:172–179, 2014.

[31] Q. Guo and L. Tang. An improved scatter search algorithm for the single machine total weighted tardiness scheduling problem with sequence-dependent setup times. *Applied Soft Computing*, 29:184–195, 2015.

[32] S. Hajforoosh, M. A. Masoum, and S. M. Islam. Real-time charging coordination of plug-in electric vehicles based on hybrid fuzzy discrete particle swarm optimization. *Electric Power Systems Research*, 128:19–29, 2015.

[33] P. Hansen, N. Mladenović, and J. Moreno Pérez. Variable neighborhood search: methods and applications. *Annals of Operations Research*, 175(1):367–407, 2010.

[34] M. Hapke and R. Slowinski. Fuzzy priority heuristics for project scheduling. *Fuzzy Sets and Systems*, 83(3):291–299, 1996.

[35] Y. He, B. Venkatesh, and L. Guan. Optimal scheduling for charging and discharging of electric vehicles. *IEEE Transactions on Smart Grid*, 3(3):1095–1105, 2012.

[36] S. Heilpern. The expected value of a fuzzy number. *Fuzzy Sets and Systems*, 47:81–86, 1992.

[37] A. Hernández-Arauzo, J. Puente, M. González, R. Varela, and J. Sedano. Dynamic scheduling of electric vehicle charging under limited power and phase balance constraints. In *Proceedings of the 7th Scheduling and Planning Applications Workshop (SPARK 2013)*, pages 1–8, 2013.

[38] A. Hernández-Arauzo, J. Puente, R. Varela, and J. Sedano. Electric vehicle charging under power and balance constraints as dynamic scheduling. *Computers & Industrial Engineering*, 85:306–315, 2015.

[39] M. Honarmand, A. Zakariazadeh, and S. Jadid. Optimal scheduling of electric vehicles in an intelligent parking lot considering vehicle-to-grid concept and battery condition. *Energy*, 65:572–579, 2014.

[40] J. Hu, A. Saleem, S. You, L. Nordstrom, M. Lind, and J. Ostergaard. A multi-agent system for distribution grid congestion management with electric vehicles. *Engineering Applications of Artificial Intelligence*, 38:45–58, 2015.

[41] X. Hu, C. Zou, C. Zhang, and Y. Li. Technological developments in batteries: A survey of principal roles, types, and management needs. *IEEE Power and Energy Magazine*, 15(5):20–31, 2017.

[42] IBM. IBM CPLEX Optimizer, 2014.

[43] E. Iversen, J. Morales, and H. Madsen. Optimal charging of an electric vehicle using a markov decision process. *Applied Energy*, 123:1–12, 2014.

[44] J. J. Palacios, M. A. González, C. R. Vela, I. González-Rodríguez, and J. Puente. Genetic tabu search for the fuzzy flexible job shop problem. *Computers & Operations Research*, 54:74–89, 2015.

[45] C. Jin, J. Tang, and P. Ghosh. Optimizing electric vehicle charging: a customer's perspective. *IEEE Transactions on Vehicular Technology*, 62(7):2919–2927, 2013.

[46] C. Jin, J. Tang, and P. Ghosh. Optimizing electric vehicle charging with energy storage in the electricity market. *IEEE Transactions on Smart Grid*, 4(1):311–320, 2013.

[47] J. Kang, S. J. Duncan, and D. N. Mavris. Real-time scheduling techniques for electric vehicle charging in support of frequency regulation. *Procedia Computer Science*, 16:767–775, 2013.

[48] S. Kaplan and G. Rabadi. Exact and heuristic algorithms for the aerial refueling parallel machine scheduling problem with due date-to-deadline window and ready times. *Computers & Industrial Engineering*, 62(1):276–285, 2012.

[49] D. Karaboga. An idea based on honey bee swarm for numerical optimization. Technical report, Erciyes university, engineering faculty, computer engineering department, 2005.

[50] D. Karaboga, B. Gorkemli, C. Ozturk, and N. Karaboga. A comprehensive survey: artificial bee colony (abc) algorithm and applications. *Artificial Intelligence Review*, 42(1):21–57, 2014.

[51] E. Karfopoulos and N. Hatziargyriou. A multi-agent system for controlled charging of a large population of electric vehicles. *IEEE Transactions on Power Systems*, 28(2):1196–1204, 2013.

[52] H. Kim, J. Lee, and G. Park. Constraint-based charging scheduler design for electric vehicles. In *Proceedings of the 4th Asian conference on Intelligent Information and Database Systems*, volume Part III, pages 266–275, 2012.

[53] M. Kuroda and Z. Wang. Fuzzy job shop scheduling. *International Journal of Production Economics*, 44:45–51, 1996.

[54] D. Lei. Solving fuzzy job shop scheduling problems using random key genetic algorithm. *International Journal of Advanced Manufacturing Technologies*, 49:253–262, 2010.

[55] B. Liu, Y. Fan, and Y. Liu. A fast estimation of distribution algorithm for dynamic fuzzy flexible job-shop scheduling problem. *Computers & Industrial Engineering*, 87:193–201, 2015.

[56] J. Lopes, F. Soares, P. Almeida, and M. Moreira da Silva. Smart charging strategies for electric vehicles: Enhancing grid performance and maximizing the use of variable renewable energy resources. In *EVS24 International Battery, Hybrid and Fuel Cell Electric Vehicle Symposium*, pages 392–396, 2009.

[57] Z. Luo, Z. Hu, Y. Song, Z. Xu, and H. Lu. Optimal coordination of plug-in electric vehicles in power grids with cost-benefit analysis – part i: Enabling techniques. *IEEE Transactions on Power Systems*, 28(4):3546–3555, 2013.

[58] C. Ma, J. Rautiainen, D. Dahlhaus, A. Lakshman, J.-C. Toebermann, and M. Braun. Online optimal charging strategy for electric vehicles. *Energy Procedia*, 73:173–181, 2015.

[59] Z. Ma, D. Callaway, and I. Hiskens. Decentralized charging control of large populations of plug-in electric vehicles. *IEEE Transactions on Control Systems Technology*, 21(1):67–78, 2013.

[60] M. J. Mirzaei, A. Kazemi, and O. Homaee. Real-world based approach for optimal management of electric vehicles in an intelligent parking lot considering simultaneous satisfaction of vehicle owners and parking operator. *Energy*, 76:345–356, 2014.

[61] N. Mladenović and P. Hansen. Variable neighborhood search. *Computers & Operations Research*, 24:1097–1100, 1997.

[62] N. Mladenović and P. Hansen. Variable neighborhood search: principles and applications. *European Journal of Operational Research*, 130:449–467, 2001.

[63] D. Oliveira, A. Zambroni de Souza, and L. Delboni. Optimal plug-in hybrid electric vehicles recharge in distribution power systems. *Electric Power Systems Research*, 98:77–85, 2013.

[64] I. Oliver, D. Smith, and J. Holland. A study of permutation crossover operators on the tsp. In *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 224–230, 1987.

[65] J. J. Palacios, I. González-Rodríguez, C. R. Vela, and J. Puente. Coevolutionary makespan optimisation through different ranking methods for the fuzzy flexible job shop. *Fuzzy Sets and Systems*, 278:81–97, 2015.

[66] J. J. Palacios, I. González-Rodríguez, C. R. Vela, and J. Puente. Robust multiobjective optimisation for fuzzy job shop problems. *Applied Soft Computing*, 56:604–616, 2017.

[67] J. Puente, C. R. Vela, and I. González-Rodríguez. Fast local search for fuzzy job shop scheduling. In *Proceedings of ECAI 2010*, pages 739–744. IOS Press, 2010.

[68] I. Rahman, P. Vasant, B. Singh, M. Abdullah-Al-Wadud, and N. Adnan. Review of recent trends in optimization techniques for plug-in hybrid, and electric vehicle charging infrastructures. *Renewable and Sustainable Energy Reviews*, 58:1039–1047, 2016.

[69] I. Rahman, P. M. Vasant, B. S. M. Singh, and M. Abdullah-Al-Wadud. Swarm intelligence-based smart energy allocation strategy for charging stations of plug-in hybrid electric vehicles. *Mathematical Problems in Engineering*, 2015(Article ID 620425):10 pages, 2015.

[70] P. Richardson, D. Flynn, and A. Keane. Optimal charging of electric vehicles in low-voltage distribution systems. *IEEE Transactions on Power Systems*, 27(1):268–279, 2012.

[71] H. Rommelfanger. FULPAL — an interactive method for solving (multiobjective) fuzzy linear programming problems. In R. Slowinski and J. Teghem, editors, *Stochastic Versus Fuzzy Approaches to Multiobjective Mathematical Programming under Uncertainty*, pages 279–299. Springer, 1990.

[72] J. Sedano, M. Portal, A. Hernández Arauzo, J. Villar, J. Puente, and R. Varela. Sistema de control autónomo para distribución de energía en una estación de carga de vehículos eléctricos: diseño y operación. Technical report, Instituto Tecnológico de Castilla y León ITCL, 2012.

[73] J. Sedano, M. Portal, A. Hernández-Arauzo, J. R. Villar, J. Puente, and R. Varela. Intelligent system for electric vehicle charging: Design and operation. *DYNA*, 88(6):644–651, 2013.

[74] A. Sheikhi, S. Bahrami, A. Ranjbar, and H. Oraee. Strategic charging method for plugged in hybrid electric vehicles in smart grids; a game theoretic approach. *International Journal of Electrical Power & Energy Systems*, 53:499–506, 2013.

[75] S.-O. Shim and Y.-D. Kim. Scheduling on parallel identical machines to minimize total tardiness. *European Journal of Operational Research*, 177(1):135–146, 2007.

[76] F. Soares, P. Rocha Almeida, and J. Peças Lopes. Quasi-real-time management of electric vehicles charging. *Electric Power Systems Research*, 108:293–303, 2014.

[77] W. Su and M.-Y. Chow. Computational intelligence-based energy management for a large-scale phev/pev enabled municipal parking deck. *Applied Energy*, 96:171–182, 2012.

[78] W. Su and M.-Y. Chow. Performance evaluation of an eda-based large-scale plug-in hybrid electric vehicle charging algorithm. *IEEE Transactions on Smart Grid*, 3(1):308–315, 2012.

[79] O. Sundstrom and C. Binding. Flexible charging optimization for electric vehicles considering distribution grid constraints. *IEEE Transactions on Smart Grid*, 3(1):26–37, 2012.

[80] T. Tran, M. Dogru, U. Ozen, and C. Beck. Scheduling a multi-cable electric vehicle charging facility. In *Proceedings of SPARK'13. ICAPS'13 Scheduling and Planning Applications woRKshop, ICAPS council*, pages 20–26, 2013.

[81] S. Vandael, B. Claessens, M. Hommelberg, T. Holvoet, and G. Deconinck. A scalable three-step approach for demand side management of plug-in hybrid vehicles. *IEEE Transactions on Smart Grid*, 4(2):720–728, 2013.

[82] C. R. Vela, R. Varela, and M. A. González. Local search and genetic algorithm for the job shop scheduling problem with sequence dependent setup times. *Journal of Heuristics*, 16:139–165, 2010.

[83] B. Wang, Q. Li, X. Yang, and X. Wang. Robust and satisfactory job shop scheduling under fuzzy processing times and flexible due dates. In *Proc. of the 2010 IEEE International Conference on Automation and Logistics*, pages 575–580, 2010.

[84] B. K. Wong and V. S. Lai. A survey of the application of fuzzy set theory in production and operations management: 1998–2009. *International Journal of Production Economics*, 129:157–168, 2011.

[85] D. Wu, D. Aliprantis, and L. Ying. Load scheduling and dispatch for aggregators of plug-in electric vehicles. *IEEE Transactions on Smart Grid*, 3(1):368–376, 2012.

[86] X. Wu, X. Hu, S. Moura, X. Yin, and V. Pickert. Stochastic control of smart home energy management with plug-in electric vehicle battery energy storage and photovoltaic array. *Journal of Power Sources*, 333:203–212, 2016.

[87] S. Xu, D. Feng, Z. Yan, L. Zhang, N. Li, L. Jing, and J. Wang. Ant-based swarm algorithm for charging coordination of electric vehicles. *International Journal of Distributed Sensor Networks*, 2013(Article ID 268942):13 pages, 2013.

[88] J. Yang, L. He, and S. Fu. An improved PSO-based charging strategy of electric vehicles in electrical distribution grid. *Applied Energy*, 128:82–92, 2014.

[89] P.-S. You and Y.-C. Hsieh. A hybrid heuristic approach to the problem of the location of vehicle charging stations. *Computers & Industrial Engineering*, 70:195–204, 2014.

[90] A. Zakariazadeh, S. Jadid, and P. Siano. Multi-objective scheduling of electric vehicles in smart distribution system. *Energy Conversion and Management*, 79:43–53, 2014.

[91] K. Zhan, Z. Hu, Y. Song, N. Lu, Z. Xu, and L. Jia. A probability transition matrix based decentralized electric vehicle charging method for load valley filling. *Electric Power Systems Research*, 125:1–7, 2015.

[92] S. Zhong, Y. Chen, and J. Zhou. Fuzzy random programming models for location-allocation problem with applications. *Computers & Industrial Engineering*, 89:194–202, 2015.

Parte II

Compendio de publicaciones

Capítulo 7

Listado de publicaciones

En este capítulo incluimos información detallada sobre la calidad de las publicaciones realizadas durante el desarrollo de esta tesis doctoral. En la Sección 7.1 presentamos las publicaciones correspondientes a revistas científicas indexadas en el Journal Citations Report (JCR), las publicaciones de la Sección 7.2 corresponden a congresos relevantes según el GII/GRIN/SCIE Conference Rating, mientras que en la Sección 7.3 incluimos otras publicaciones presentadas en congresos menos relevantes.

7.1. Publicaciones en revistas indexadas en el Journal Citations Report (JCR)

1. García-Álvarez, J.; González, M.A.; Vela, C.R. Metaheuristics for solving a real-world electric vehicle charging problem. *Applied Soft Computing* **2018**, 65, 292–306. DOI: 10.1016/j.asoc.2018.01.010.

   Factor de impacto (JCR 2018): 4.873

   Factor de impacto (5 años): 4.858

   Ranking:

   - Computer Science, Interdisciplinary Applications: 11/106 **(Q1)**
   - Computer Science, Artificial Intelligence: 20/133 **(Q1)**

2. García-Álvarez, J.; González-Rodríguez, I.; Vela, C.R.; González, M.A.; Afsar, S. Genetic fuzzy schedules for charging electric vehicles. *Computers & Industrial Engineering* **2018**, 121, 51–61. DOI: 10.1016/j.cie.2018.05.019.

   Factor de impacto (JCR 2018): 3.518

   Factor de impacto (5 años): 3.600

   Ranking:

   - Computer Science, Interdisciplinary Applications: 24/106 **(Q1)**
   - Engineering, Industrial: 11/46 **(Q1)**

3. García Álvarez, J.; González, M.A.; Rodríguez Vela, C.; Varela, R. Electric vehicle charging scheduling by an enhanced artificial bee colony algorithm. *Energies* **2018**, 11, 2752. DOI: 10.3390/en11102752.

Factor de impacto (JCR 2018): 2.707

Factor de impacto (5 años): 2.990

Ranking:

- Energy & Fuels: 56/103 **(Q3)**

7.2. Publicaciones en congresos relevantes según el GII/GRIN/SCIE Conference Rating

1. García-Álvarez, J.; González, M.A.; Vela, C.R. A genetic algorithm for scheduling electric vehicle charging. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2015)*. Madrid, **July 2015**, 393–400. DOI: 10.1145/2739480.2754695.

GGS Class: 2

GGS Rating: A

7.3. Publicaciones en otros congresos

1. García-Álvarez, J.; González, M.A.; Vela, C.R. A GRASP approach to solve an electric vehicle charging scheduling problem. *The XI Metaheuristics International Conference (MIC 2015)*. Agadir, **June 2015**, 1–10.

2. García-Álvarez, J.; González-Rodríguez, I.; González, M.A.; Vela, C.R. Planificación genética de la carga de vehículos eléctricos bajo incertidumbre. *Actas de la XVII Conferencia de la Asociación Española para la Inteligencia Artificial (MAEB 2016)*. Salamanca, **September 2016**, 93–102.

3. García-Álvarez, J.; González, M.A.; Vela, C.R.; Varela, R. Electric vehicle charging scheduling using an artificial bee colony algorithm. *Proceedings of the International Work-Conference on the Interplay Between Natural and Artificial Computation (IWINAC2017)*. A Coruña, **June 2017**, 115–124. DOI: 10.1007/978-3-319-59740-9_12.

Capítulo 8

Publicaciones en revistas indexadas en el Journal Citations Report (JCR)

En este capítulo presentamos las publicaciones correspondientes a revistas científicas indexadas en el Journal Citations Report (JCR) que dan cuerpo a la tesis doctoral.

8.1. Metaheuristics for solving a real-world electric vehicle charging problem

Presentamos la siguiente publicación:

- **Título:** Metaheuristics for solving a real-world electric vehicle charging problem.
- **Revista:** *Applied Soft Computing*.
- **Año:** 2018.
- **DOI:** 10.1016/j.asoc.2018.01.010.
- **Referencia:** García-Álvarez, J.; González, M.A.; Vela, C.R. Metaheuristics for solving a real-world electric vehicle charging problem. *Applied Soft Computing*. 2018, 65, 292–306. DOI: 10.1016/j.asoc.2018.01.010.

Author's copy of
J. Garcia-Alvarez, M.A. Gonzalez, C. R. Vela,
Metaheuristics for solving a real-world electric vehicle charging scheduling problem,
Applied Soft Computing (2018)
DOI of published final version: 10.1016/j.asoc.2018.01.010

Metaheuristics for solving a real-world electric vehicle charging scheduling problem

Jorge García-Álvarez[a], Miguel A. González[a,*], Camino R. Vela[a]

[a]*Department of Computing, University of Oviedo. Campus of Gijón, 33204, Gijón, Spain.*

Abstract

In this paper we consider a problem motivated by a real-world environment: scheduling the charging periods for a large set of electric vehicles, subject to a set of hard constraints, with the objective of minimizing the total tardiness. The set of constraints imposed by the charging station makes it difficult to obtain schedules that are both feasible and efficient. We consider both the static variant of the problem, where arrival time, charging time and due date of vehicles are known in advance, and also the dynamic variant. As the problem is NP-hard, metaheuristics are probably the best option to solve it. In fact, the state-of-the-art methods are a simple genetic algorithm (*GA*) and a method based on priority rules (*EVS*). In this paper we propose to design hybrid metaheuristics, motivated by the success of these hybridizations in solving a large number of scheduling problems. In particular we define a GRASP-like method and a memetic algorithm that use the Variable Neighborhood Search framework, both specifically designed for the problem at hand. Experimental results illustrate the potential of the proposed methods, reaching in some test-beds improvements larger than 12% with respect to *EVS* and *GA*.

Keywords: scheduling, electric vehicle, charging strategy, GRASP, memetic algorithm

1. Introduction

During the last decades, environmental impact due to carbon footprint has led to an increased interest in the research and development of electric vehicle (EV) technologies [22]. Additionally, the use of EVs reduces the dependency on petrol, as the electricity they need can be generated by a number of different sources, including renewable ones [58]. Furthermore, in [2] and [27] it is pointed out that EV's may contribute to efficient use of energy, as they may be used as auxiliary sources of energy storage to compensate for fluctuations in the generation of renewable energy, and also as a way of regulating the power of the network and voltage profiles. Over these last years there is a huge amount of research about many different aspects of EVs, as comfortable driving strategies [1], energy-efficient routing [34] or vehicle-to-grid scheduling [50], to cite just a few examples.

One of the problems of EVs is that the use of large fleets of them may overload the grid due to the high charging time of the batteries [23]. In fact, developing smart charging systems that avoid peak demands is considered as one of the challenges in EV technology [4]. Whenever possible, vehicles should be charged at times of low consumption in order to balance the electrical needs. There are many approaches in the literature that try to fill the overnight valley with the objective to reduce costs (see for example [10], [33] and [57]). However, calculating a good schedule that minimizes costs and/or maximizes user satisfaction is usually difficult because of the particular physical constraints of each charging station or the limited amount of energy available.

*Corresponding author. Phone number: +34 985182493
Email addresses: jgarcia-alvarez@outlook.com (Jorge García-Álvarez), mig@uniovi.es (Miguel A. González), crvela@uniovi.es (Camino R. Vela)

Our work considers the EV charging scheduling problem presented in [19, 20]. This is a problem inspired by a real charging station in which EVs require charging while they are stationed in their private parking spaces. The station has some technological restrictions, for example the power supply is limited and so there is a maximum number of vehicles that can charge simultaneously. Moreover, the energy must be balanced between the three lines of the three-phase electric power source. These constraints make it difficult to schedule the charging times in such a way that all customers are satisfied, and therefore a smart control system is needed, as it is pointed out in [45].

In [20] this problem is defined and solved using a problem decomposition approach, based on decomposing the overall problem in three subproblems, one per line of the three-phase electric feeder. These three subproblems are modelled as instances of the single machine scheduling problem with variable capacity, and they are solved with a procedure based on dispatching rules, considering as optimization criteria the minimization of the total tardiness. Afterwards, in [12] the authors study the same problem and propose a genetic algorithm with operators specifically designed for it. In addition to the real-world dynamic problem, they also study the static one in which the arrival times, charging times and due dates of all vehicles are known in advance. This variant of the problem is also relevant for several reasons: for example it provides reference values that can be useful to compare the relative performance of algorithms in solving the dynamic problem.

As we will see in Section 2, there are many different models of EV charging scheduling problems. Depending on the requirements of each specific charging station, the set of constraints or even the objective functions may differ. The main difference of the model considered here with respect to other models in the literature is that it considers the imbalance between lines as a hard constraint, whereas other models consider it as an additional objective to be minimized, or even it is not considered at all. Another distinctive feature is the fact that each vehicle must use its own charging point located in a particular line, and therefore balancing the vehicles between the three lines may be hard. These characteristics make it difficult to build schedules that are feasible and efficient at the same time.

The problem outlined is NP-hard ([20]) and therefore metaheuristics are recommended in order to obtain good results in a reasonable computational time. In fact, in Section 2 we review a number of metaheuristic approaches from the literature that solve similar problems. In this paper we define two new approaches: the first is inspired by the Greedy Randomized Adaptive Search Procedure (GRASP) [43]. In our proposal, the construction phase incorporates a local refinement procedure to fix some inefficient charging time assignments that may appear due to the maximum imbalance constraint. For the improvement phase we define four different neighborhood structures and study its combinations in order to use the Variable Neighborhood Search (VNS) metaheuristic. We also limit the size of the neighborhoods by considering only those vehicles that need to be scheduled earlier. Our second approach is a memetic algorithm inspired by the genetic algorithm proposed in [12], which is specifically designed to solve this scheduling problem. The algorithm is enhanced with a hybrid crossover operator that helps preserving the diversity of the population. Moreover, its results are further improved by combining it with the Variable Neighborhood Search metaheuristic.

In summary, we confront a problem inspired in an industrial case, not just an academic one. For this reason, algorithms must be specifically designed for the problem at hand and, in principle, they are not suitable to solve other EV scheduling problems. Moreover, the state of the art for it are the methods from [12] and [20], which are rather simple metaheuristics. The motivation of our work is that metaheuristics usually improve their results when hybridized with local search [11, 40, 56]. The contribution of the paper is twofold. Firstly, we propose competitive methods to find feasible and efficient solutions to this problem; the methods were specifically designed for solving it. Secondly, this work constitutes the first use of local search techniques and hybrid algorithms to address this particular problem. Concretely, the definitions of neighborhood structures for the VND strategy are original to this paper.

The remainder of the paper is organized as follows. First of all we review some recent papers in Section 2. After defining the problem in Section 3, we propose in Section 4 the GRASP and the memetic algorithm. To evaluate our proposals, in Section 5 we present detailed results analyzing the performance of the different components of our algorithms and we also compare them with the state of the art in a set of real-world inspired instances. Finally, Section 6 presents some conclusions.

2. Literature review

In recent years a number of EV charging scheduling models have been proposed in the literature, which proves the interest of the research community. In [13] and [41] we can find comprehensive reviews of optimization methods for EV charging infrastructure.

Strategies for EV charging can be divided into two categories [33]. In decentralized or distributed strategy, individual EV owners have authority to make decisions about the time and rate of charging of their EVs. The network operator can impose some price incentives in order to move charging tasks to valleys of the load profile, as it is done for example in [3]. Although this strategy offers more flexibility to EV owners, it may not ensure optimality in the charging of EVs [33] [55]. Moreover, it can produce some security concerns in the power grid. On the other hand, in centralized charging strategies, an aggregator centrally makes the scheduling decisions about the time and rate of charging of EVs to try to obtain the optimal solution. Therefore, the EV can be plugged in by its owner at any time, but the starting and ending time of the charging task is remotely decided by the aggregator, who tries to benefit both the customers and the distribution company at the same time [25]. To find the optimal solution, the aggregator receives real-time status of loads and EVs' battery state-of-charge along with their arrival and desired departure times specified by EV owners [8]. Most authors propose the centralized strategy, although we can also find some examples of decentralized control methods in [7], [59] and [62].

Regarding the objectives, some works try to minimize the total cost [18], while others try to not only maximize the profit of the parking operator but also the satisfaction of vehicle owners [35]. Many authors try to minimize peak demands and grid congestion by filling the valleys of power availability [9] [10] [29] [33] [53] [62]. Another interesting objective is to minimize the deviation between the energy bought in the market and the energy consumed by EVs [49]. When using a three-phase power flow, minimizing the load imbalance between phases is often considered as an objective [30]. As we can see, there are many interesting objectives, and hence some works consider multi-objective EV charging problems. For example, in [61] both total operational costs and emissions are minimized. In [27] a real-time EV charging scheduling system is proposed that tries to optimize total load variance and owners' preferences at the same time. Also, in [6] the authors coordinate the charging of single-phase EVs with dynamic behavior in unbalanced three-phase distribution systems employing smart grid facilities. Two objective functions are minimized: the total cost of purchasing energy in a multi-tariff pricing environment and also grid total energy losses over charging span. Another example can be found in [16], where the authors minimize the costs associated with energy generation and grid losses while maximizing the power delivered to the vehicles, considering distribution transformer loading, voltage regulation limits, and initial and final battery state-of-charge based on consumers' preferences.

Constraints also vary, as different charging stations and environments lead to different sets of constraints and considerations. Distribution network technical constraints and EVs' power demand are considered by almost all authors. Some also consider varying electricity prices [15] [32] [47] [52], or varying charging rate of the vehicles in each time slot [9] [10]. In [7] EVs' users can even adapt their charging rate according to their preferences. Other authors also introduce uncertainty and stochastic parameters to optimize the charging strategy [24] [31], or even other considerations such as desired charging electricity price or age of the battery [21], or the possibility of using EVs to store energy in order to mitigate the impact of uncertainty [26]. Moreover, in some charging stations the scheduler can decide to which dock each vehicle is assigned [54], while in other charging stations, as the one considered in this paper, each user is the owner of a particular dock and therefore he must charge his EV in that dock.

In the literature we can find different metaheuristic techniques for tackling these problems, for example particle swarm optimization [60], artificial immune systems [38], gravitational search algorithm [42], estimation of distribution algorithm [51] [52], ant-based swarm algorithm [59] or fuzzy genetic algorithm [16]. Other types of algorithms include linear programming [44], moving window optimization [32], game theory based optimization [47] or a distributed multi-agent method based on the Nash Certainty Equivalence Principle [28].

It is worth mentioning that the methods reported in this literature review solve different EV charging scheduling problems, as different charging stations have their specific sets of constraints and objective

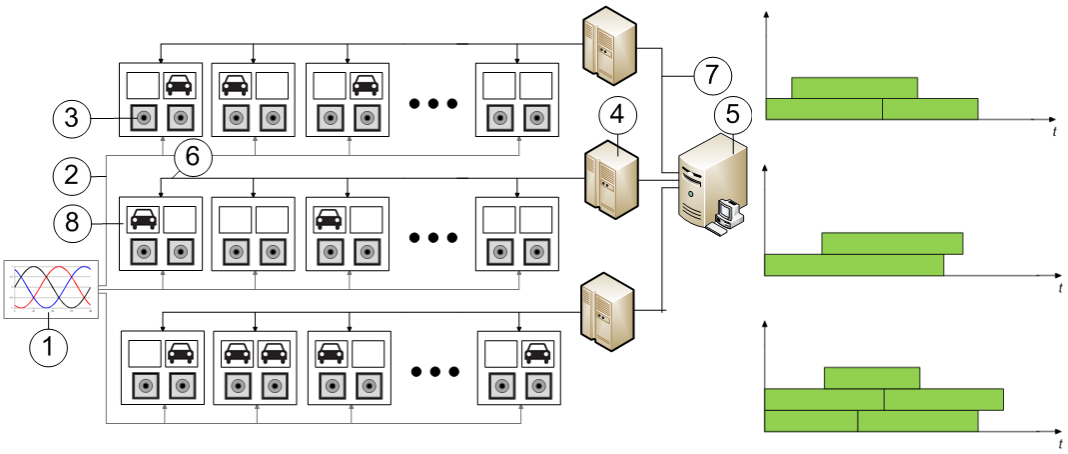


Figure 1: Distribution net of the considered charging station. (1) power source, (2) three-phase electric power, (3) charging points, (4) masters, (5) server with database, (6) communication RS 485, (7) communication TCP/IP, (8) slaves. The Gantt Chart depicted indicates the charging intervals of the vehicles in each line.

functions. Therefore, they cannot naturally be compared with the methods presented in our work.

3. Problem definition

The distribution net of the charging station was designed to be installed in a private parking where each client has his own parking space [46]. In Figure 1 we show its main components. There is a three-phase power source, and each particular parking space has a charging point which is connected to one single-phase and, when it is active, it transfers energy at a constant rate (230v and 2.3 Kw). The parking has 180 spaces in total, 60 connected to each line.

There is a distributed system that controls the station, whose main components are:

- A main server with a database that stores all data about vehicles and charging schedules.
- A number of masters in each line, which gather information from the slaves and send connection/disconnection signals to them based on the data read from the database.
- One slave for every two consecutive charging points in the same line. The slaves activate and deactivate the charging points, and they also record asynchronous events, like a new vehicle arriving to the station.

When a user enters the station he/she has to input the charging time and the due date for taking the vehicle away. With these data the control system must create a schedule (i.e. establish the starting time for the charging of each vehicle) so that the total tardiness is minimized.

In order to create a charging schedule for the vehicles the system must take into account some constraints. There is a limited contracted power, and so there is a maximum number of vehicles that can be charging at the same time in any given line. Moreover, the power consumed by the three lines must be balanced at all times, because of economical and electro-technical reasons. It is well-known that an imbalanced system causes higher losses and lowers the energy transmission efficiency. Also, there are recent regulations in Spain (BOE, 22 September 2013) that do not permit highly imbalanced systems unless there is the consentient of the supplier company, and even in that case the customer can be penalized. Another constraint is that each user is the owner of his own parking space, and therefore his/her vehicle must be connected to a particular charging point in a given line. Notice that this fact combined with the imbalance constraint makes the problem really hard, as we cannot avoid imbalances by distributing the vehicles between lines at will. An

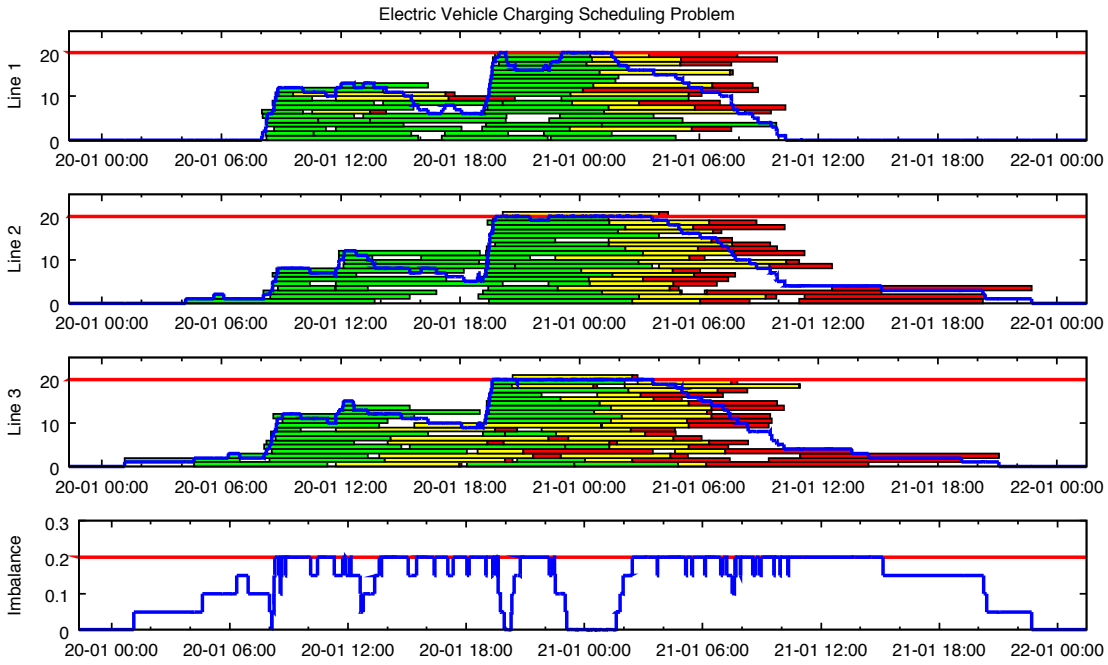


Figure 2: Example of a problem schedule. Charging intervals of vehicles that finish charging before their due date are colored in green, whereas charging intervals of vehicles with tardiness are colored in yellow and red (in red the portion after the due date, which corresponds to the tardiness of the vehicle). For each line we mark in blue its occupation in each particular time point, and at the bottom of the figure it is reported the maximum imbalance level among the lines.

added difficulty is that the charging of a vehicle cannot be interrupted. In [46] more details about the constraints of the real charging station are given.

In the real environment, the arrival and charging times of the vehicles are not known in advance, and hence it is a dynamic problem. However, in [12] it is pointed out that the static version of the problem is also interesting for several reasons. For instance, when having the solution to the static problem we can assess how much we can still improve the solution to the dynamic problem. Moreover, it may give hints regarding how a good solution to the dynamic problem should look, and therefore give some ideas to efficiently solve it. For these reasons, in this paper we will also study both variants. In the following we present the formulation of the problem, which is that presented in [20].

Figure 2 shows an example of a schedule for one of the instances of the benchmark described in Section 5.1, with 180 vehicles evenly distributed in each of the three lines. The X-axis corresponds to the time, whereas the Y-axis represents the occupation of each particular line (marked in blue in each line). The charging intervals are drawn by rectangles, colored in green if the corresponding EV finishes its charging on time, and in yellow and red otherwise (using red color for the portion of the charging interval exceeding its due-date, which corresponds to the tardiness of the vehicle). We also show at the bottom of the figure the maximum imbalance level between the lines in each time point. When that level reaches 0.4 it means that there are 8 EVs of difference between two of the lines, which is the maximum allowed with the parameters of the considered instance, i.e. $N = 20$ and $\Delta = 0.4$ (see Section 3.1).

3.1. The static problem

In the static version of the problem we have all data regarding the arrival times of the EVs, their charging times and due dates. Its formulation is as follows: we consider three charging lines $L_i, 1 \leq i \leq 3$, each one having n_i charging points. There can be a maximum of $N > 0$ active charging points at the same time in any given line. A total of M_i vehicles arrive to the line L_i, and these vehicles are denoted by $v_{i1}, \ldots, v_{iM_i}$.

For each vehicle v_{ij} we are given its arrival time $t_{ij} \geq 0$, its charging time $p_{ij} > 0$ and the time at which the user is expected to take the vehicle out of the charging station, or due date, $d_{ij} \geq t_{ij} + p_{ij}$. Ideally, the battery of the EV should be fully charged by the due date, however this may not be possible and so some tardiness may be incurred.

The goal of this scheduling problem is to obtain a feasible schedule that minimizes the total tardiness. A feasible schedule is an assignment of starting times to the decision variables st_{ij} for each vehicle v_{ij}, $1 \leq i \leq 3$, $1 \leq j \leq M_i$, such that the following constraints are all satisfied:

1. Each vehicle must start charging after its corresponding arrival time.

$$\forall v_{ij}, \quad st_{ij} \geq t_{ij} \tag{1}$$

2. A vehicle v_{ij} can not be disconnected before its charging time C_{ij}, i.e. no preemption is allowed.

$$C_{ij} = st_{ij} + p_{ij} \tag{2}$$

3. There can be a maximum of N active charging points in any line at a given time.

$$\max N_i(t) \leq N, \quad t \geq 0; 1 \leq i \leq 3 \tag{3}$$

where $N_i(t)$ denotes how many active charging points are in line L_i at time t.

4. There is a maximum imbalance between any two lines L_i and L_j, which is specified by the parameter Δ.

$$\max \left(\frac{|N_i(t) - N_j(t)|}{N} \right) \leq \Delta, \quad t \geq 0; 1 \leq i, j \leq 3 \tag{4}$$

The objective function to be minimized is the total tardiness, defined as follows:

$$\sum_{i=1}^{3} \sum_{j=1}^{M_i} \max(0, C_{ij} - d_{ij}) \tag{5}$$

The schedule depicted in Figure 2 is feasible and its total tardiness is 223.56 hours.

3.2. The dynamic problem

The dynamic problem may be seen as a sequence $P_1, P_2, \ldots, P_n$ of instances of a static problem. Each instance P_k is composed by some vehicles that have arrived to the charging station but have not yet started to charge, and some vehicles that are already charging. Formally, in an instance P_k at a time T_k we consider a set of vehicles $\{v_{i1}, \ldots, v_{ia_i}, \ldots, v_{im_i}\}$ in each line of the charging station L_i, $1 \leq i \leq 3$. For each vehicle v_{ij} we have its charging time p_{ij} and its due date d_{ij}.

Some vehicles $(v_{i1}, \ldots, v_{ia_i})$ have already started to charge and have not yet finished, i.e. for all vehicles $1 \leq j \leq a_i$ it holds that $st_{ij} < T_k$ and it holds that $C_{ij} = st_{ij} + p_{ij} > T_k$. On the other hand, the remaining vehicles $v_{ia_i+1}, \ldots, v_{im_i}$ have entered the facility but have not yet started to charge.

In an instance P_k, the capacity of line L_i to charge new vehicles at a given time t, denoted $M_i^k(t)$, can be calculated as follows

$$M_i^k(t) = N - \sum_{j=1}^{a_i} X_{ij}(t), \quad t \geq T_k \tag{6}$$

where

$$X_{ij}(t) = \begin{cases} 1, & \text{if } t < C_{ij} \\ 0, & \text{if } t \geq C_{ij} \end{cases} \tag{7}$$

The goal is to obtain a feasible schedule for all unscheduled vehicles that are at the station at time T_k. Hence, we have to assign a starting time st_{ij}^* for each one of them that satisfies all constraints derived from the static problem. Again, the objective function is the minimization of the total tardiness.

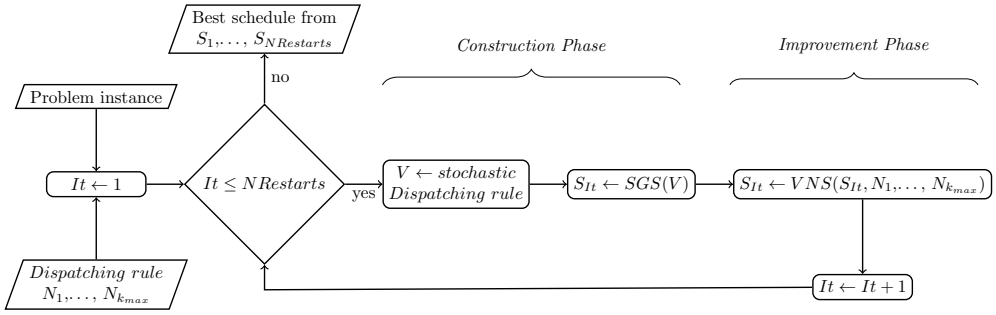


Figure 3: Flow chart of the GRASP-like algorithm.

As we have seen, solving the dynamic problem consists basically in solving a sequence of static problems, an approach already presented in [12] and [20]. Ideally, a new schedule should be built each time a new vehicle that requires charging arrives to the station. However, in the real setting new schedules are calculated at most at time intervals of length ΔT (fixed at two minutes), in order to not overload the system if many vehicles arrive in a short period of time.

At each time point T_k a supervisor program which is running on the server checks if any vehicle has arrived since the last time point T_{k-1}. In that case, a new instance P_k is created and solved, and the new solution replaces the current one. On the other hand, if no vehicle has arrived in the last time interval, the current solution remains valid for an additional ΔT time. Notice that we can assign different st^*_{ij} to a single vehicle in different P_k instances as long as that vehicle does not start charging. In any case, as soon as it starts charging, its st^*_{ij} cannot be modified anymore.

4. Solving the problem with metaheuristics

Metaheuristic techniques allow problem-specific information to be incorporated and exploited, as well as to deal with complex objectives, in particular with "messy real world objectives and constraints" [5]. In Section 4.1 we propose a solving method inspired by the GRASP metaheuristic [43], whereas in Section 4.2 we describe a memetic algorithm.

4.1. A GRASP-like algorithm

Our GRASP-like method implements a construction phase followed by an improvement phase. As soon as a feasible solution is built in the construction phase, the second phase performs a local search where the solution's neighborhoods are systematically explored until a local optimum is found. Some random decisions are present, hence the process is repeated $NRestarts$ times and the final output is the best solution from all runs. Figure 3 shows the outline of the GRASP-like algorithm described in the following.

4.1.1. Construction Phase

In the construction phase we use a Sequential Greedy Stochastic algorithm (SGS) to build an initial schedule, that is, an initial assignment of starting times for all EVs such that all constraints hold.

SGS is in fact a schedule builder that exploits a dispatching rule. We consider two of these rules: the Due Date rule (DDR) and the Apparently Tardiness Rule (ATR). The first one simply sorts all vehicles in ascending order based on the due date d_{ij}, whereas the second exploits a rule of common use in tardiness minimization (see for example [12], [20] and [48]).

We are using ATR as follows: let $\Gamma(\alpha)$ the earliest possible starting time for a vehicle not yet scheduled in a partial schedule α. Then, we calculate a selection probability for all unscheduled vehicles that are able to start charging at $\Gamma(\alpha)$ in the following way:

$$\Pi_j = \frac{1}{p_j} \exp\left[\frac{-\max(0, d_j - \Gamma(\alpha) - p_j}{g\overline{p}}\right] \tag{8}$$

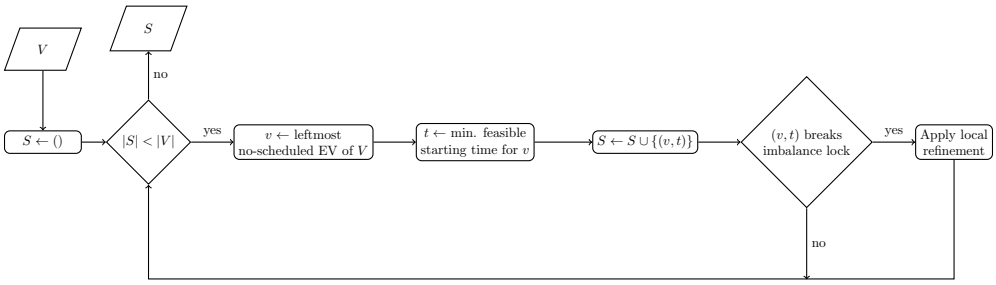


Figure 4: Flow chart of the SGS algorithm.

where $\overline{p}$ is the average charging time of the vehicles and g is a look-ahead parameter. We fix $g = 0.25$, as suggested in [20].

These heuristics are able to build efficient solutions, but we cannot use deterministic versions of them because we need to build different solutions. Hence we are using stochastic versions in the following way: to select the following vehicle to be appended to the priority list we sort all vehicles according to the corresponding rule and then we perform a tournament selection (i.e. we select $tSize$ random vehicles and append the best of them according to the corresponding rule). Notice that $tSize$ is a relevant parameter, as small values lead to nearly random solutions, whereas large values may produce too similar solutions. This parameter is fixed empirically in Section 5.

As soon as we have the priority list, V, we can apply the SGS to obtain its corresponding solution. To this end, it iterates over the list assigning the lowest possible starting time to each vehicle such that all constraints hold. It is worth mentioning that it is always possible to schedule an EV holding all constraints, as in the worst case we can assign to it as starting time the maximum completion time of all the previously scheduled EVs. That is, unless we define a maximum of zero vehicles in a given line and/or a maximum of zero imbalance between lines, which are not realistic assumptions. Therefore, it is always possible to produce a feasible schedule provided that $N > 0$ and $\Delta \geq \frac{1}{N}$. It is also important to remark that SGS solves the complete problem, not each line separately as is done in [20].

However, as pointed out in [12], the constraint of maximum imbalance when using a priority list may lead to some poor assignments of starting times for charging vehicles. In that paper a local refinement procedure is proposed that tries to improve the partial schedule in the following way: when we assign a starting time for charging a vehicle, we check if it breaks a previous imbalance lock, defined as follows.

Definition 1. *A feasible partial schedule has an imbalance lock in an interval $[t_1, t_2)$ if, and only if, there exist two lines L and L' such that no more EVs can be scheduled in L' in the interval $[t_1, t_2)$ because of the maximum imbalance constraint (see Equation 4) with respect to line L. Formally, $\forall t' \in [t_1, t_2)$ it holds that:*

$$N_{L'}(t') < N, \tag{9}$$

$$\frac{N_{L'}(t') + 1 - N_L(t')}{N} > \Delta. \tag{10}$$

Definition 2. *A feasible assignment of a starting time t for an EV v of a line L with charging time p, breaks a previous imbalance lock in the interval $[t_1, t_2)$, where $t \leq t_1 < t_2 \leq t + p$, if and only if before the assignment there exists an imbalance lock in the interval $[t_1, t_2)$ between lines L, L' (i.e. conditions 9 and 10 hold) but after the assignment a new EV can be scheduled in line L' in this interval. Formally, $\forall t' \in [t_1, t_2)$ it holds that:*

$$\frac{N_{L'}(t') + 1 - N_L(t') + 1}{N} \leq \Delta, \tag{11}$$

$$\frac{N_{L'}(t') + 1 - N_{L''}(t')}{N} \leq \Delta, \tag{12}$$

where $L \neq L' \neq L''$.

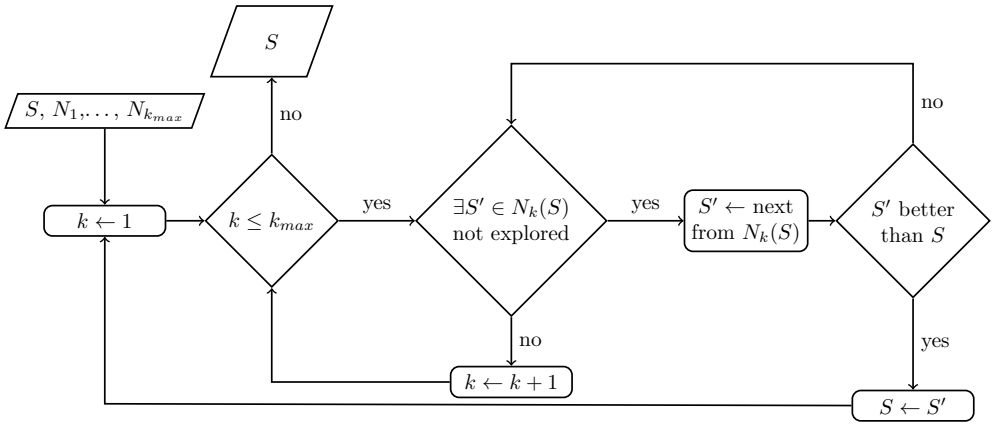


Figure 5: Flow chart of the VNS algorithm.

If this situation occurs, we unschedule all EVs from lines L' and L'' such that they start charging at a time greater than or equal to t_2. Hence, they may be later rescheduled in an earlier starting time. In [12] some examples of this procedure are given.

Finally, when we obtain the full schedule, the initial priority list is rebuilt so that it reflects the final order in which the vehicles are scheduled. The flow chart of the SGS algorithm is shown in Figure 4. In Section 5 we report some experiments to compare the results of our GRASP algorithm when using this local refinement procedure with those obtained when not using it.

4.1.2. Improvement Phase

The improvement phase implements a local search that starts from a feasible solution provided by SGS and tries to improve its objective value, i.e. the total tardiness.

As selection/replacement strategy we use a simple hill climbing. Therefore, as soon as we evaluate a neighbor that improves the current solution, the move is accepted, the neighbor replaces the current solution, and the process starts again; otherwise the move is discarded. The hill climbing continues until no replacement has been made for a whole round of trials, i.e. a local optimum has been reached.

As neighborhood structures, we consider several variants of the classical insertion and swap neighborhoods and we analyze their combinations in order to use Variable Neighborhood Search (VNS). This is a metaheuristic proposed by Mladenović and Hansen [36], which is based on the idea of systematically changing the neighborhood. Its advanced principles for solving combinatorial optimization problems and applications were further introduced in [37] and recently in [17].

The swap neighborhood consists in interchanging pairs of elements. To generate the neighborhood of a solution, we iterate over the list of vehicles starting by the first vehicle of the list. We consider moving those vehicles that need to be scheduled earlier, trying to swap them with vehicles in the same line that allow for some delay. Specifically, for each vehicle v_i with positive tardiness, we consider swapping it with a vehicle v_j in the same line without tardiness and such that its starting time is lower than that of v_i.

The insertion neighborhood is similar, but instead of swapping vehicles v_i and v_j in the list, we insert v_i just before v_j.

As we use a simple hill climbing strategy, the order in which the neighbors are generated is clearly relevant. For this reason, we also consider the swap and insert neighborhoods that iterate over the priority list starting from the last vehicle of the list, therefore leading to four different neighborhoods.

After the generation of each neighbor, the SGS has to be launched to reschedule the new neighbor and calculate its objective value. However, notice that the first part of a schedule does not change from a given solution to a neighbor. To save computational time, the neighbors are not rescheduled from scratch; instead the SGS starts from the first vehicle in the list modified by the move.

To combine the defined neighborhoods with the VNS metaheuristic, we use the variable neighborhood descent (VND) approach, which is probably the least computationally expensive variant of VNS, although still very effective (see for example [14]). It is implemented by defining a list of neighborhoods N_1, N_2, ..., $N_{k_{max}}$. Basically the idea is to start performing a hill climbing with the first neighborhood. When a local optimum is reached with the current neighborhood we change to the next neighborhood, and every time we improve the current solution we return back to the first neighborhood. The algorithm finishes when we reach a solution which is a local optimum with respect to all considered neighborhoods. The flow chart of the VNS algorithm can be seen in Figure 5.

4.2. A memetic algorithm

The proposed memetic algorithm is based on that proposed in [12], being the main differences the scheduling algorithm, the crossover operator and the additional local search step. At first, we generate an initial population that contains random and heuristic solutions. The algorithm then performs a number of generations until a stopping criterion is met. In each one, the next population is built from the previous one by applying genetic operators (selection, crossover, mutation and replacement). The algorithm ends after a number of consecutive generations without improving the best solution, or if the population contains a solution with zero tardiness. Finally, we apply local search to some of the best chromosomes of the final population, to further improve them. In the following we detail the components of the memetic algorithm, whose flow chart is depicted in figure 6.

To codify chromosomes we have decided to use permutations of the vehicles. In order to build a schedule S from a permutation V, we simply schedule all vehicles of V sequentially. For each vehicle we select the earliest possible starting time such that all constraints are met with respect to previously scheduled vehicles. Clearly, all permutations represent feasible solutions (see Section 4.1.1). In this case we are not using the local refinement procedure described in Section 4.1.1 for the GRASP, as we have experimentally observed that the additional overload does not allow the memetic algorithm to evolve properly and its overall performance is worse when using it.

It is critical for the initial population to contain good and diverse solutions. To this end, we propose to create some random chromosomes (i.e. random permutations of vehicles) together with some heuristic chromosomes built using the stochastic version of the ATR and DDR rules presented in Section 4.1.1. One third of the initial population is created with each technique.

For selection and replacement we borrow the strategies presented in [14]. In each generation, firstly all chromosomes from the current population are randomly grouped into pairs. Then the crossover operator is applied to each pair of parents with some probability P_c in order to produce two offspring solutions, and then the mutation operator is applied to each offspring. The purpose of the replacement strategy is to choose the solutions that will form the next population. In particular, for each group of two parents and its two offspring, the best two solutions with different tardiness will be chosen. In [14] the authors prove that this strategy preserves the diversity of the population and is able to improve the results of the algorithm. Also notice that the best solution in a given population is always better than or equal to that of the previous population. In our problem, this means that the best assignment of starting times for EVs in a given generation of the algorithm has always a total tardiness lower than or equal to that of the best assignment of the previous generation.

The crossover operator combines some already existing solutions to generate new solutions, in such a way that they inherit relevant characteristics of the parent solutions, and have the chance to improve their objective function (i.e. the total tardiness). We are considering two different crossover operators that combine two parent solutions to generate two offspring solutions:

- The *SBX* (Starting-time Based Crossover) operator was proposed in [12] to solve this particular charging scheduling problem. The operator first selects a random time point t_0, and then a first offspring is generated taking all vehicles of the first parent that start its charging before t_0, and completed with the remaining vehicles using the relative order they have in the second parent. Another offspring may be created by simply swapping the role of the parents.

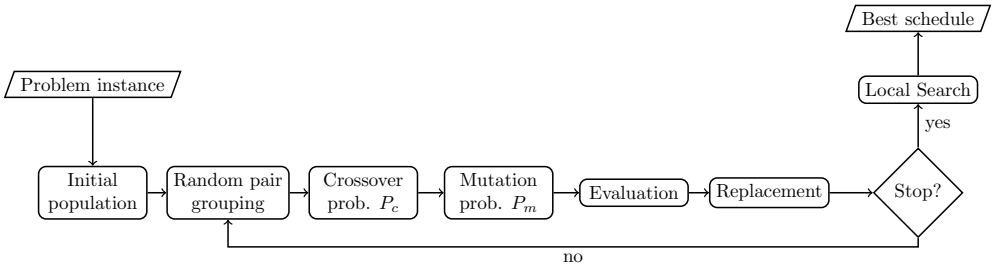


Figure 6: Flow chart of the memetic algorithm.

- The *CX* (Cycle Crossover) operator was first proposed in [39] and it is well-known for its good performance in genetic algorithms that use a permutation-based representation. Basically, it identifies a number of "cycles" between two parent chromosomes. Then, to create the first offspring, it copies the first cycle from the first parent, the second cycle from the second parent, the third cycle from the first parent, and so on. To create a second offspring the parents reverse their roles.

We refer the interested reader to the original papers for detailed examples of these operators. We have selected *SBX* and *CX* because in a preliminary study we have confirmed that they provide better results than *LEX* [12], or the well-known *PMX* and one point crossover operators. Moreover, we propose to combine the two operators in such a way that the first offspring is created with *SBX* and the second offspring is created with *CX*. We denote this hybrid operator by *SBX-CX*. We have empirically confirmed in Section 5.3 that this combination produces better results than each operator alone, probably because of an increased diversity in the population.

The purpose of a mutation operator is to introduce new genetic material in the population by doing random modifications in some chromosomes. Hence it is useful for improving the diversity of the population. We are using the operator described in [12], which mutates each one of the three lines in which EVs will be charged with some probability P_m. To mutate a line L_i, a random subset of consecutive vehicles of that line is selected, and its order is randomly shuffled.

Genetic algorithms are often combined with local search in order to provide additional intensification in the search, which usually improves their results (see for example [14]). The main concern here is that evaluating a solution is computationally costly, and hence applying the local search procedure described in Section 4.1.2 to every generated chromosome is prohibitive. We tried different strategies, and the configuration that yielded the best results was to only apply the local search to some chromosomes of the last generation of the memetic algorithm, in particular to the 5% best chromosomes such that they have different tardiness value (see Section 5.3).

5. Experimental Analysis

We have conducted an experimental study to evaluate our proposals. We start this section describing the benchmark considered. Then, we perform a parameter tuning to choose the best values for the parameters of our algorithms and finally we compare their results with those obtained by the state of the art approaches.

Our methods are implemented in C++ using a single thread and experiments were carried out on a Xeon E5520 running Linux (SL 6.0). Given its stochastic nature, the algorithms are run 30 times to obtain statistically significant results.

Throughout this section we will do some statistical tests to analyze differences between the different configurations. Since we have multiple-problem analysis, we used non-parametric statistical tests. First, we run a Shapiro-Wilk test to confirm the non-normality of the data. Then we used paired Wilcoxon signed rank tests to compare the average tardiness values obtained by the different configurations. In these tests, the level of confidence used was 95% and the alternative hypothesis was "the difference between the average

Table 1: Comparison between a standard scheduler and the improved scheduler. The values represent average tardiness (in hours) grouped by instance type.

	ATR		DDR	
Type	Standard	Improved	Standard	Improved
1	58.3	**55.2**	57.8	**55.2**
2	1140.5	**1120.6**	1138.2	**1118.4**

tardiness values is smaller than 0". The p-values obtained with these tests will determine if the differences are statistically significant or not.

5.1. Considered benchmark

In this study we consider the real-world inspired benchmark proposed in [20]. It corresponds to a charging station with 180 spaces (60 in each of the three lines), and the profiles of arrival times, due dates and demands are based on the behavior of real users. The time horizon is one day and three different scenarios are considered: scenario 1 represents a normal weekly day, with some vehicles arriving all day long, but with some arrival peaks. On the other hand, scenarios 2 and 3 represent a more extreme situation where most vehicles arrive almost at the same time. The difference between scenarios 2 and 3 is that the last one has tighter due dates.

Also, there are two types of instances with different distribution of vehicles on the lines. In type 1 instances 60 vehicles arrive at each line along the day, whereas in type 2 instances 108 vehicles arrive at L_1, 54 at L_2 and 18 at L_3. Clearly, type 2 instances promote imbalance situations among the lines, and hence it may be more difficult to obtain a satisfactory solution for them. Four different values for the imbalance parameter Δ are considered (0.2, 0.4, 0.6 and 0.8), and three values for the maximum number of vehicles N that can be charging at the same time in a given line (20, 30 and 40).

There are 30 instances for each of the 72 possible combinations of scenario, type, Δ and N, yielding 2160 instances. See [20] for more details and also the probability distributions used to generate these instances. The benchmark is openly available on the web[1]. There are no other benchmarks for this problem; remember that we are dealing with a problem based in an industrial case, not an academic one.

We have to remark that in the parameter tuning (Sections 5.2 and 5.3) we did not consider the full benchmark of 2160 instances. Instead, we only considered a set of 24 instances from scenario 1, in particular the first one from each combination of (type, Δ, N).

5.2. Parameter tuning and analysis of the GRASP algorithm

Firstly we analyze the behavior of the different versions of the GRASP algorithm, considering the two defined dispatching rules in the SGS module, with different tournament sizes, with and without the local refinement procedure in the scheduler, and trying the different local searches proposed. Notice that the best configuration for the static and dynamic problems could differ.

We start this study by looking for the best configuration for solving the static problem. Firstly we did some experiments to select the most appropriate tournament size for each dispatching rule, considering sizes 2, 5, 8 and 11. In these experiments, the number of restarts for GRASP was $NRestarts = 25$. The tournament size that achieved the best average tardiness is 2 for both ATR and DDR. In the case of DDR the differences are statistically significant (p-values are 0.02318 against size 5, 0.02318 against size 8 and 0.006794 against size 12). However, in the case of ATR the differences are not statistically significant (p-values are 0.1054 against size 5, 0.05116 against size 8 and 0.06308 against size 12). Anyway, we select size 2 for both heuristics from now on, as it appears to be the best value.

Then we made experiments to assess the performance of the local refinement procedure described in Section 4.1.1. In Table 1 we show a comparison between a standard scheduler (i.e. a scheduler that simply iterates over the permutation of vehicles and schedules each vehicle as soon as possible) and the

[1]Repository section in `http://www.di.uniovi.es/iscop`

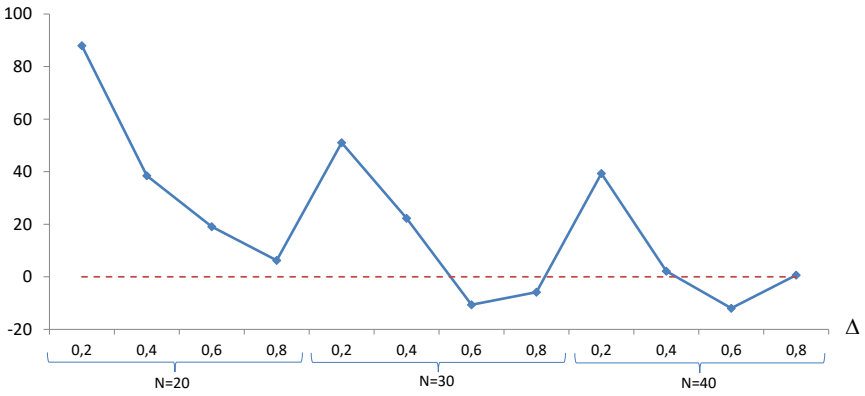


Figure 7: Difference between total tardiness values (in hours) reached when using the local refinement procedure in the scheduler and when not using it, in 12 type 2 instances with different values of Δ and N.

improved scheduler with the local refinement procedure. To obtain similar running times, we parameterized $NRestarts = 50$ when using the standard scheduler and $NRestarts = 25$ when using the improved scheduler. It can be seen that the improved scheduler obtains a better average result in every case. The statistical tests return a p-value of 0.004126 in the DDR case, and 0.01851 in the ATR case. Therefore we can conclude that the improvement provided by the local refinement procedure is statistically significant and hence we are using it in all subsequent experiments.

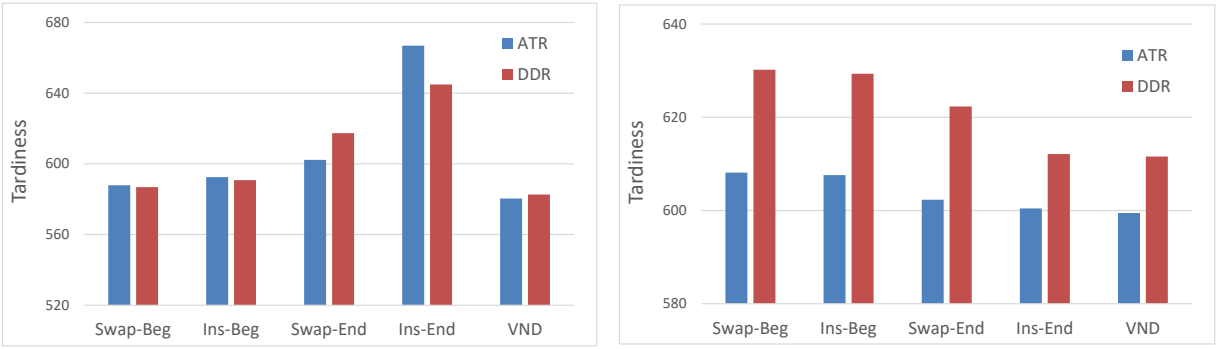
A more fine-grained analysis reveals the strong dependence between the instance parameters and the contribution of the local refinement procedure to the overall quality of solutions. Figure 7 shows the comparison between using the local refinement procedure and not using it in 12 type 2 instances. As expected, the improvement obtained decreases when the value of parameter Δ increases, and for a given value of Δ, the improvement decreases when N increases. This behavior may be due to the local reassignment being less relevant if the constraints are less restrictive, as in this case the basic scheduling algorithm is more likely to assign efficient charging intervals. In fact, in some of the less constrained instances we can see that the basic scheduling algorithm is able to outperform the advanced scheduler.

Figure 8(a) shows a comparison between the four proposed local search strategies, i.e. swap or insertion iterating from the beginning or the end of the permutation. We report the average tardiness (in hours) obtained in all considered instances. The best neighborhood appears to be $Swap - Beg$, then $Ins - Beg$, then $Swap - End$ and finally $Ins - End$. The p-values obtained in the statistical tests show that all these differences are statistically significant for both ATR and DDR. We also show the results of the Variable Neighborhood Descent algorithm using the two best performing neighborhoods, i.e. $N_1 = Swap - Beg$ and $N_2 = Ins - Beg$. To obtain similar running times, we parameterized $NRestarts = 25$ when using a single neighborhood structure and $NRestarts = 20$ when using the VND approach. It can be seen that the best results are those of the VND approach; in fact, the performance when using VND is significantly better than when using the best neighborhood alone (p-value of 0.0002895 in the ATR case and 0.0001155 in the DDR case). Hence we can conclude that VND is significantly better than a standard hill climbing approach.

Finally, if we compare the VND approach using DDR and ATR, the latter obtains a better average result, but the differences are not statistically significant (p-value of 0.2283).

Overall, we conclude that the best configuration for GRASP in the static problem is using the ATR dispatching rule with a tournament size of 2, the scheduler with the local refinement procedure, and a VND approach in the improvement phase with $N_1 = Swap - Beg$ and $N_2 = Ins - Beg$.

To find the best configuration for the dynamic version of the problem, we followed the same procedure as in the static version. Therefore, we started by selecting the most appropriate tournament size for each dispatching rule. Again, the tournament size that achieves the best average tardiness is 2 for both dispatching rules and, again, the differences are statistically significant for DDR but not for ATR. Afterwards, we have confirmed that in the dynamic problem the local refinement procedure also improves the results of the



(a) Static problem

(b) Dynamic problem

Figure 8: Comparison between different local search strategies.

scheduling algorithm, and this improvement is statistically significant.

In Figure 8(b) we show a comparison of the local search strategies. In this case the best neighborhood appears to be $Ins - End$, then $Swap - End$, then $Ins - Beg$ and finally $Swap - Beg$, and the p-values obtained in the statistical tests show that all these differences are statistically significant for both ATR and DDR, except in the case of neighborhoods $Ins - Beg$ and $Swap - Beg$ for the DDR case, where the differences are not statistically significant (p-value of 0.3755). Notice that the best neighborhoods in the dynamic version of the problem are those that performed the worst in the static version. It seems that iterating the list from the first position is better in the static case but worse in the dynamic case. The reason may be related to the size of the problems, as in the static version we must consider all vehicles at once, whereas in the dynamic version we usually have to deal with a small number of vehicles.

Overall, the configuration that achieves the best average tardiness values is a Variable Neighborhood Descent metaheuristic using the two best neighborhoods: $N_1 = Ins - End$ and $N_2 = Swap - End$. However, although the average results are better, in this case the differences are not statistically significant with respect to using a simple hill climbing with only the best neighborhood $Ins - end$ (for example, the p-value in the ATR case is 0.09918). It is remarkable that in the dynamic problem, the ATR heuristic is much better than the DDR heuristic, unlike what happens in the static problem, where their results are similar. In fact, if we compare DDR and ATR when using the VND approach, we can conclude that ATR is significantly better (p-value of 0.0006193).

In summary, the best configuration for GRASP in the dynamic problem is using the ATR dispatching rule with a tournament size of 2, the scheduler with the local refinement procedure and a VND approach with $N_1 = Ins - End$ and $N_2 = Swap - End$. Therefore, the only change with respect to the static problem are the chosen neighborhood structures.

5.3. Parameter tuning and analysis of the memetic algorithm

As our memetic algorithm is inspired by the genetic algorithm proposed in [12], we take as base configuration the parameter setting reported in that paper: a population size PS of 200 chromosomes, the stopping condition set at 25 generations without improving the best solution found so far (or also if we find a solution with zero tardiness), crossover probability P_c of 0.8, mutation probability P_m of 0.1 for each line of the schedule, and tournament size $tSize$ of 8 for the dispatching rules used to create the initial population.

For the local search we propose to use the best configuration found in the previous section for $GRASP$; i.e. a VND approach with $N_1 = Swap - Beg$ and $N_2 = Ins - Beg$ for the static problem, and with $N_1 = Ins - End$ and $N_2 = Swap - End$ for the dynamic problem. Using the presented parameter setting the computational time is adequate (see Section 5.5), in fact it is slightly lower than that of the GRASP approach.

Furthermore, this configuration results in reasonable convergence patterns, as shown in Figure 9, which details the evolution of the best and average tardiness of the population for one run on an instance of

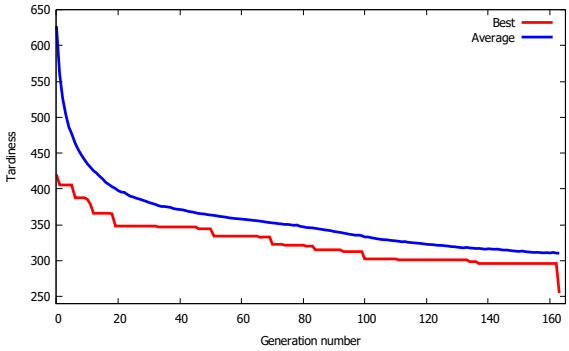
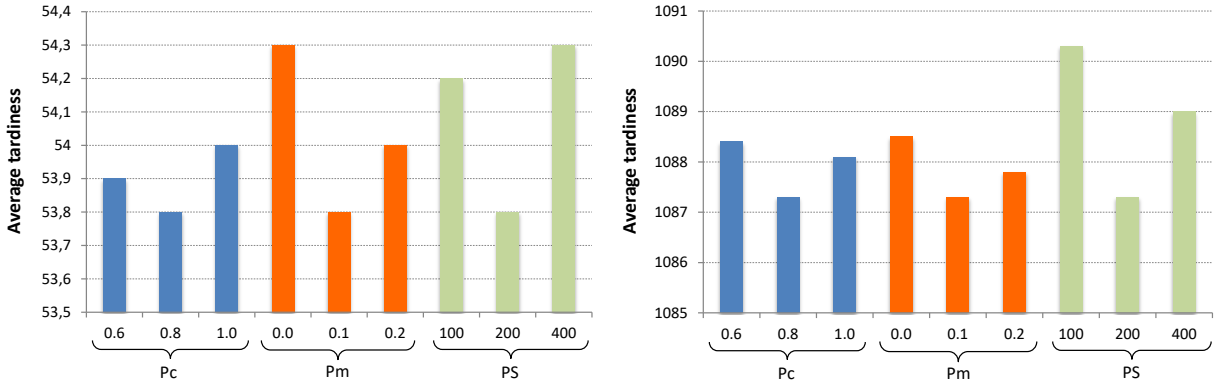


Figure 9: Evolution of the best and average tardiness of the population depending on the generation number, for one run on an instance of scenario 1, type 1, $N = 20$ and $\Delta = 0.2$. The remarkable improvement in the last generation is due to the application of the local search.



(a) Type 1 instances

(b) Type 2 instances

Figure 10: Comparison between different configurations of the memetic algorithm.

scenario 1, type 1, $N = 20$ and $\Delta = 0.2$. It can be seen the improvement obtained in the last generation by using the Variable Neighborhood Descent metaheuristic.

As the presented memetic algorithm has some extra characteristics with respect to the genetic algorithm proposed in [12], we have to ensure that the base configuration chosen is in fact the best possible configuration. Therefore, in the following we report the results of some experiments modifying the crossover probability (P_c), mutation probability (P_m) and population size (PS) with respect to the base configuration. Figures 10(a) and 10(b) report the results achieved in the static problem in type 1 and type 2 instances, respectively. We see that a crossover probability of 0.8 obtains better results than those of 0.6 and 1.0 in both types of instances. Moreover, the differences are statistically significant (p-value of 0.0001959 against 0.6 and 0.008232 against 1.0). Regarding the mutation probability, $P_m = 0.1$ presents lower average tardiness values than $P_m = 0.0$ or $P_m = 0.2$. In this case the differences are statistically significant against no mutation (p-value of 0.0001016) but not statistically significant against 0.2 probability (p-value of 0.1119). Then, we have compared population sizes 100, 200 and 400, also modifying the stopping condition to ensure that the running times are similar and so the comparison is fair. A population size of 200 chromosomes obtains the best results, and the differences are both statistically significant (p-value of 3.206e-05 against 100 and 0.04762 against 400). In summary, the base configuration taken from [12] is slightly better than other alternatives for the memetic algorithm.

Another relevant decision is choosing how many chromosomes we should improve with local search. Table 2 reports the results achieved in the static problem by applying the local search to a different percentage

Table 2: Comparison of tardiness and running times between applying local search to a different percentage of chromosomes of the final population of the memetic algorithm. The results are averaged, grouped by instance type.

	0%		5%		100%	
Type	Tard.	T(s)	Tard.	T(s)	Tard.	T(s)
1	58.0	12.9	53.8	28.1	53.0	177.2
2	1092.9	38.6	1087.3	80.5	1085.9	330.7

Table 3: Comparison of crossover operators for the memetic algorithm. The values represent average tardiness (in hours) grouped by instance type.

Type	SBX	CX	SBX-CX
1	56.7	54.1	**53.8**
2	1157.4	1094.7	**1087.3**

of solutions of the final population of the memetic algorithm. Obviously, the best results are obtained by improving all chromosomes, but the huge increase in running time does not compensate for the slight, although statistically significant improvement in tardiness values (p-value of 0.0002897 against 5% and 4.778e-05 against 0%). On the other hand, the improvement obtained by applying the local search to the 5% best chromosomes is also statistically significant (p-value of 4.778e-05 against 0%) and in this case it may be worth the reasonable increase in running time. Moreover, we have tried to run the standard genetic algorithm for as much time as this configuration and the results were still worse. For these reasons we propose 5% as the best alternative. The results in the dynamic problem are similar, although the differences in running times are smaller due to the fact that the subproblems are usually small and so the local search is in average much faster.

Finally, we report in Table 3 some results that prove the superiority of the hybrid operator *SBX-CX* over *SBX* and *CX*. It can be seen that it obtains better average tardiness in both instance types, and in fact the differences are statistically significant (p-value of 1.647e-07 against *SBX* and 3.9e-05 against *CX*).

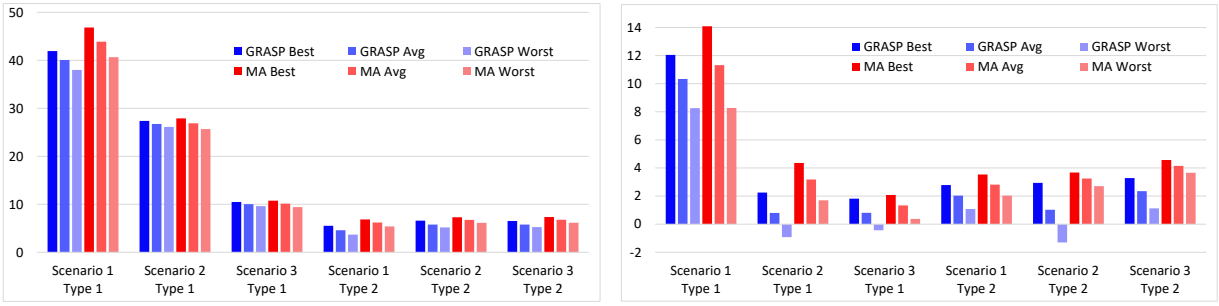
5.4. Comparison with the state of the art

Tables 4, 5 and 6 show the comparison between the best configuration of our algorithms and the state of the art in all instances of scenarios 1, 2 and 3 respectively. For GRASP, we parameterized $NRestarts = 25$ in all these experiments. The results are grouped in sets of 30 instances that depend on the instance parameters (type, Δ and N). The values correspond to the sum of the tardiness of all 30 instances of each group. Column GA reports the average values obtained by the genetic algorithm proposed in [12] for scenario 1 (the authors do not report results for scenarios 2 and 3). Column EVS shows the tardiness values obtained by the dispatching rule based procedure proposed in [20] for the dynamic problem in all three scenarios. Columns $GRASP$ and MA report the average results of our GRASP-like algorithm and memetic algorithm respectively. We mark in bold the best value for each version of the problem.

As expected, the results in the static version of the problem are much better than those of the dynamic version. In particular, for $GRASP$ the average solution in the static problem is better than that of the dynamic problem in 2037 instances, equal in 72 instances and worse in 51, whereas in the case of MA it is better in 2077 instances, equal in 72 instances and worse in 11 instances. The p-values are in both cases lower than 2.2e-16 and therefore the differences are statistically significant. We can conclude that the algorithms are able to exploit all the extra information about arrival times, charging times and due dates in order to reach better solutions.

Regarding the comparison between our results for the dynamic version and those of EVS [20], the average result of $GRASP$ is better in 1539 instances, equal in 75 instances and worse in 546 instances, whereas the average result of MA is better in 1827 instances, equal in 73 instances and worse in 260 instances. In both cases, p-values lower than 2.2e-16 confirm that our methods are significantly better than EVS.

About the comparison between our methods and the genetic algorithm proposed in [12] (GA), in the static version of the problem the average results of $GRASP$ are better in 385 instances, equal in 126 and



(a) Static problem

(b) Dynamic problem

Figure 11: Percentage improvement of $GRASP$ and MA with respect to EVS.

worse in 209, whereas in the dynamic version they are better in 305 instances, equal in 110 and worse in 305. In the static version of the problem $GRASP$ is significantly better than GA (p-value of 4.473e-12), whereas in the dynamic version of the problem it occurs the opposite and GA is significantly better than $GRASP$ (p-value of 0.03244). About our memetic algorithm, in the static version of the problem MA is better than GA in 575 instances, equal in 125 and worse in 20, whereas in the dynamic version it is better in 408 instances, equal in 110 and worse in 202. In this case, p-values lower than 2.2e-16 confirm that in both cases MA is significantly better than GA. In fact, in the static problem the average improvement is larger than 12%.

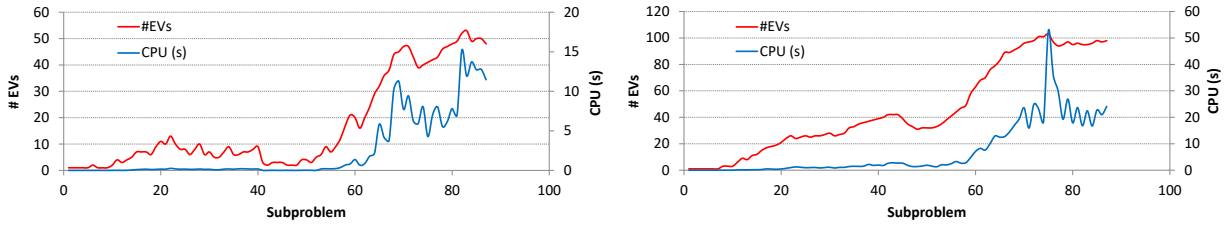
Finally, if we compare the results of $GRASP$ with those of MA, in the static problem the average result of MA is better than that of $GRASP$ in 1785 instances, equal in 195 and worse in 180, whereas in the dynamic problem it is better in 1318 instances, equal in 155 and worse in 687. In both cases the p-value is lower than 2.2e-16, proving that MA is significantly better than $GRASP$. However, we can notice in Tables 4, 5 and 6 that $GRASP$ performs slightly better than MA in some groups of instances.

Figures 11(a) and 11(b) show the average percentage improvement of our methods with respect to the results reported in [20], in the static and dynamic problem respectively. The greatest improvements are those obtained in type 1 instances considering scenario 1. Also, the improvements in the static version of the problem (more than 40% in some cases) are much greater than those obtained in the dynamic version (up to 14%). It is also remarkable the small difference between our best and worst results, which proves the robustness of the methods.

We can also notice in Tables 4, 5 and 6 the differences in tardiness depending on the instance parameters. It is much easier to find solutions with low total tardiness in type 1 instances, as in this case the load on all lines is even, whereas in type 2 instances 60% of the EVs go to line L_1, 30% to line L_2 and only 10% to line L_3. The difficulty of the instances also depends on the parameters N and Δ, and as expected they are more difficult as these values become smaller, i.e. the constraints are more strict. In the most extreme case we can find instances of type 2, $N = 20$ and $\Delta = 0.2$, where each EV may have average tardiness values over 24 hours, which are very large considering that the time horizon is one day. This fact can be explained because as soon as all EVs (only 18) that go to line L_3 finish their charging, the number of EVs that can charge at the same time on any of the other lines is limited to only 4 due to the strict imbalance constraint. Hence, a bottleneck is created that produces huge delays, specially in line L_1, where 90 EVs must be charged. This particular combination of parameters does not reflect a real situation, but those instances are also useful to test the ability of the scheduling algorithms to manage extreme cases.

5.5. Computational cost considerations

As our algorithms are intended to run in a real-time scenario, it is important to discuss their computational cost. In the dynamic problem we have observed that the time depends on the instance parameters and on the number of vehicles of the particular subproblem. Overall, the average running time of MA was 224 seconds per run. These times represent the total time for solving the whole problem. This is the measure of computational cost considered in [12], where it is reported an average of 325 seconds per run for GA. As



(a) Type 1, $N = 20$, $\delta = 0.2$ (b) Type 2, $N = 20$, $\delta = 0.2$

Figure 12: CPU time and number of EVs in each subproblem of a dynamic instance.

the authors use the same machine as in this work, we can conclude that the running time of MA is about 31% lower than that of GA.

In order to know if MA is appropriate in the real-time setting, we have to check if the CPU time required to solve any subproblem of the dynamic version is lower than the 120 seconds between consecutive subproblems. The required time heavily depends on the number of EVs to be scheduled and on the type of the problem. Figure 12 illustrates this fact showing one run in two of the hardest instances: (a) Type 1, $N = 20$, $\Delta = 0.2$ and (b) Type 2, $N = 20$, $\Delta = 0.2$. We report the CPU time required to solve each subproblem (in blue) and the number of EVs scheduled in each one (in red). In both figures, X-axis shows the sequence of subproblems, whereas the primary Y-axis refers to the number of EVs and the secondary Y-axis shows CPU time in seconds. We can appreciate the strong dependencies mentioned. In the instance of Figure 12(a) 87 subproblems are solved, the average number of EVs in each subproblem is 17, the average CPU time is 2.41 seconds, and the largest CPU time required to solve a subproblem is 15.19 seconds in a subproblem with 53 EVs. In the instance of Figure 12(b) 90 subproblems are solved with average size of 48 EVs, the average time is 7.98 seconds and the largest is 52.84 seconds in a subproblem with 103 EVs. As these times are still far from the 120 seconds between consecutive subproblems, we can conclude that MA is appropriate in the real-time setting. We also have to remark that the time required by the algorithm EVS proposed in [20] is significantly lower, as in no case it takes more than one second to solve a subproblem. This is no surprise, as a method based on dispatching rules typically requires less time than sophisticated metaheuristics, but also obtains worse results.

Regarding the time required to solve the static problem, MA takes 67 seconds in average, whereas the algorithm GA proposed in [12] takes 93 seconds in average, and so in this case MA uses an average of 28% less computational time than GA.

Finally, we noticed that the computational times of $GRASP$ are slightly larger than those of MA in both the static and dynamic problems, even though its results are generally worse, confirming again the superiority of MA.

6. Conclusions

We have tackled a problem inspired by a real-world scenario which consists in assigning starting times to charge a set of electric vehicles in order to minimize the total tardiness, subject to a number of hard constraints. We have considered two variants of the problem: static, in which we know in advance the arrival time, charging time and due date of every vehicle, and dynamic, where we do not have these data. Even though the static variant is not realistic, its study is relevant as it may provide insight to solve the dynamic variant. We have proposed two methods specifically designed to solve our problem: one based on the GRASP metaheuristic that consists in a constructive phase followed by an improvement phase, and a memetic algorithm. We have defined four different neighborhoods for the local search, and we have considered both a hill climbing strategy and a Variable Neighborhood Descent metaheuristic. Our algorithms are comprehensively analyzed in the experimental study and then compared with the state of the art, obtaining significantly better results than previous approaches.

Table 4: Comparison with the state of the art in scenario 1 instances. Each value represents the sum of the tardiness (in hours) of the 30 instances of each group. **Bold** numbers indicate the best value for each version of the problem.

		Static problem			Dynamic problem			
N	Δ	GA [12]	GRASP	MA	EVS [20]	GA [12]	GRASP	MA
Type 1 instances								
20	0.2	5442.3	5348.9	**5210.8**	8386.3	7141.6	7182.9	**6981.2**
20	0.4	2680.1	2651.3	**2509.2**	4120.4	3976.7	3918.2	**3884.2**
20	0.6	2299.7	2320.3	**2252.7**	3670.6	3568.8	**3556.1**	3556.7
20	0.8	2238.7	2332.7	**2199.0**	3590.9	3501.9	**3496.6**	3506.3
30	0.2	996.9	983.2	**729.5**	1959.3	**1411.7**	1481.3	1444.8
30	0.4	92.1	82.2	**52.5**	421.2	374.7	**355.8**	367.6
30	0.6	50.0	36.2	**34.3**	347.9	318.9	**306.5**	316.2
30	0.8	49.2	38.1	**33.8**	347.6	316.8	**304.5**	314.7
40	0.2	364.1	350.8	**216.9**	735.0	**511.0**	546.6	541.6
40	0.4	0.4	**0.0**	**0.0**	14.0	6.4	**5.7**	7.7
40	0.6	**0.0**	**0.0**	**0.0**	**3.4**	**3.4**	**3.4**	**3.4**
40	0.8	**0.0**	**0.0**	**0.0**	**3.4**	**3.4**	**3.4**	**3.4**
Averages		1184.5	1178.6	**1103.2**	1966.7	1761.3	1763.4	**1744.0**
Type 2 instances								
20	0.2	124380.0	124565.0	**122615.0**	128185.0	124599.3	125226.8	**124075.0**
20	0.4	45263.1	44675.0	**43991.1**	46319.3	45461.3	45407.4	**45127.2**
20	0.6	21205.6	21025.6	**20526.9**	22966.8	22074.6	22089.9	**21822.6**
20	0.8	13031.0	13053.2	**12762.8**	14573.1	14337.1	14392.7	**14209.9**
30	0.2	71129.0	70682.9	**69610.6**	72860.8	71462.5	71371.7	**70894.5**
30	0.4	20629.5	20456.0	**20031.7**	21479.9	21321.0	21311.3	**21130.3**
30	0.6	7188.0	7204.1	**7030.5**	8088.9	8006.8	7984.5	**7921.5**
30	0.8	3607.7	3567.8	**3527.9**	4486.3	4407.8	4391.4	**4389.3**
40	0.2	45216.0	44658.3	**43981.8**	46135.4	45455.4	45416.1	**45139.2**
40	0.4	10010.6	10009.9	**9760.7**	10869.3	10799.0	10802.5	**10654.6**
40	0.6	2916.8	2886.9	**2851.1**	3599.1	3517.8	**3505.9**	3515.6
40	0.8	922.8	892.4	**872.3**	1635.5	1568.8	**1554.6**	1572.1
Averages		30458.3	30306.4	**29796.9**	31766.6	31084.3	31121.2	**30871.0**

In the empirical study we have seen that the evolutionary strategy produces better results than the multi-start strategy, which suggests that evolving a population of solutions for a number of iterations is better than simply improving a set of random or heuristic solutions. We can also conclude that hybrid metaheuristics are efficient methods, as the local search was able to improve the final results obtained by the genetic algorithm. About the local search, the performance of the Variable Neighborhood Descent metaheuristic is better than that of a standard hill climbing, as long as the neighborhood structures are adequately selected. Additionally, we have seen that the benefits of using an advanced schedule builder (i.e. with the local refinement procedure) are relevant when using a multi-start strategy like GRASP, but not so much when using a evolutionary metaheuristic.

We believe that the main reasons for the efficiency of our approaches, in particular the memetic algorithm, are the adequate balance between diversification and intensification in the search. The diversification is achieved by using several methods to generate good and diverse initial solutions, the hybrid crossover operator and the replacement strategy, whereas the intensification is provided by a combination of efficient neighborhood structures using the Variable Neighborhood Descent metaheuristic.

Some directions for future work are to design more efficient algorithms incorporating problem specific knowledge, and to consider more realistic energy models. For example, the possibility that the EVs may be charged at a non constant rate [10], or that the energy costs vary depending on the time of day [32].

Table 5: Comparison with the state of the art in scenario 2 instances. Each value represents the sum of the tardiness (in hours) of the 30 instances of each group. **Bold** numbers indicate the best value for each version of the problem.

		Static problem		Dynamic problem		
N	Δ	GRASP	MA	EVS [20]	GRASP	MA
Type 1 instances						
20	0.2	**14015.5**	14411.5	16886.0	16772.7	**16176.3**
20	0.4	11798.4	**11703.6**	14131.2	14429.1	**13915.9**
20	0.6	11392.1	**11328.3**	13648.3	13941.3	**13559.1**
20	0.8	11300.1	**11242.1**	13547.9	13817.9	**13492.7**
30	0.2	3936.4	**3920.6**	6394.7	5995.3	**5821.5**
30	0.4	2745.4	**2726.2**	4824.3	4716.3	**4688.4**
30	0.6	2621.1	**2592.8**	4668.6	**4546.4**	4560.3
30	0.8	2627.7	**2591.4**	4672.6	**4544.0**	4559.0
40	0.2	820.2	**638.0**	1951.4	1626.4	**1593.5**
40	0.4	190.6	**186.4**	1212.7	**1096.2**	1102.0
40	0.6	174.9	**172.2**	1210.7	**1098.8**	1103.3
40	0.8	177.0	**171.9**	1210.6	**1099.2**	1102.7
Averages		5149.9	**5140.4**	7029.9	6973.6	**6806.2**
Type 2 instances						
20	0.2	138943.0	**137204.0**	143883.0	142949.9	**139192.0**
20	0.4	58014.0	**57453.4**	62246.2	61157.7	**59194.0**
20	0.6	34846.1	**34548.2**	39869.4	36887.1	**36298.4**
20	0.8	27861.8	**27735.1**	30887.2	29955.0	**29791.0**
30	0.2	84357.5	**82970.0**	86392.0	87569.1	**84470.0**
30	0.4	32928.7	**32714.9**	34475.4	34501.4	**33936.8**
30	0.6	17195.6	**17077.3**	19355.7	18723.6	**18709.8**
30	0.8	12405.9	**12364.5**	14375.1	**14020.1**	14201.7
40	0.2	58178.9	**57652.0**	59775.0	60705.5	**58714.1**
40	0.4	20910.2	**20786.9**	22254.0	22115.0	**21984.2**
40	0.6	9637.2	**9582.4**	10991.3	**10691.2**	10815.9
40	0.8	5816.4	**5806.0**	7260.5	**7020.8**	7178.5
Averages		41757.9	**41324.6**	44313.7	43858.0	**42873.9**

Moreover, we plan to consider new objective functions, as minimizing the imbalance between the lines or the peak consumption, in addition to total tardiness minimization. Thus, modeling and solving the problem in the frameworks of multi or many objective optimization will be required.

Acknowledgments

This research has been supported by the Spanish Government under grant TIN2016-79190-R.

References

[1] D. Chakraborty, W. Vaz, and A. K. Nandi. Optimal driving during electric vehicle acceleration using evolutionary algorithms. *Applied Soft Computing*, 34:217–235, 2015.

[2] D. Dallinger. *Plug-in Electric Vehicles Integrating Fluctuating Renewable Electricity*. Kassel University Press, 2013.

[3] D. Dallinger and M. Wietschel. Grid integration of intermittent renewable energy sources using price-responsive plug-in electric vehicles. *Renewable and Sustainable Energy Reviews*, 16(5):3370–3382, 2012.

[4] EDSO. Position paper on electric vehicles charging infrastructure. Technical report, European Distribution system Operators for Smart Grids (EDSO), 10 April 2012.

Table 6: Comparison with the state of the art in scenario 3 instances. Each value represents the sum of the tardiness (in hours) of the 30 instances of each group. **Bold** numbers indicate the best value for each version of the problem.

		Static problem		Dynamic problem		
N	Δ	GRASP	MA	EVS [20]	GRASP	MA
Type 1 instances						
20	0.2	**18918.6**	19209.6	20704.7	20537.6	**20242.4**
20	0.4	16737.5	**16628.5**	18001.7	18083.8	**17916.8**
20	0.6	16293.4	**16218.8**	**17528.2**	17613.2	17573.4
20	0.8	16189.4	**16136.2**	**17463.3**	17542.7	17524.5
30	0.2	**7764.2**	7791.9	9051.1	8752.9	**8569.1**
30	0.4	6549.4	**6528.0**	7347.3	7290.4	**7283.5**
30	0.6	6404.8	**6379.6**	7150.4	**7080.3**	7115.3
30	0.8	6399.8	**6371.6**	7144.5	**7071.4**	7106.9
40	0.2	2504.9	**2422.5**	3478.1	3122.8	**3106.7**
40	0.4	1862.7	**1853.4**	2373.1	**2318.3**	2328.6
40	0.6	1825.1	**1810.3**	2276.7	**2225.8**	2247.6
40	0.8	1826.9	**1810.2**	2276.7	**2224.1**	2246.7
Averages		8606.4	**8596.7**	9566.3	9488.6	**9438.5**
Type 2 instances						
20	0.2	131321.0	**129339.0**	138064.0	132960.7	**131037.0**
20	0.4	57272.0	**56672.0**	62988.8	59611.2	**58225.8**
20	0.6	37418.1	**37089.8**	42528.3	39366.3	**38574.6**
20	0.8	31860.4	**31715.6**	33998.2	33931.6	**33360.3**
30	0.2	80438.3	**79182.4**	83169.5	82459.7	**80526.7**
30	0.4	33313.1	**33084.2**	34799.7	34680.0	**34045.1**
30	0.6	19697.9	**19592.6**	21321.1	20985.8	**20645.6**
30	0.8	15852.8	**15812.5**	**16972.0**	17150.0	16979.7
40	0.2	56339.3	**55832.1**	58306.3	58082.5	**56737.6**
40	0.4	22064.6	**21932.0**	22988.7	23091.7	**22710.2**
40	0.6	11520.4	**11462.9**	12220.3	12292.9	**12168.1**
40	0.8	8400.8	**8388.2**	**9127.3**	9265.2	9204.6
Averages		42124.9	**41675.3**	44707.0	43656.5	**42851.3**

[5] A. T. Ernst, H. Jiang, M. Krishnamoorthy, and S. D. Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research*, 153:3–27, 2004.

[6] M. Esmaili and A. Goldoust. Multi-objective optimal charging of plug-in electric vehicles in unbalanced distribution networks. *International Journal of Electrical Power & Energy Systems*, 73:644–652, 2015.

[7] Z. Fan. A distributed demand response algorithm and its application to phev charging in smart grids. *IEEE Transactions on Smart Grid*, 3(3):1280–1290, 2012.

[8] J. Foster and M. Caramanis. Optimal power market participation of plug-in electric vehicles pooled by distribution feeder. *IEEE Transactions on Power Systems*, 28(3):2065–2076, 2013.

[9] L. Gan, U. Topcu, and S. Low. Optimal decentralized protocol for electric vehicle charging. In *Proceedings IEEE Conference on Decision and Control and European Control Conference CDC-ECE*, pages 5798–5804. IEEE, 2011.

[10] L. Gan, U. Topcu, and S. Low. Optimal decentralized protocol for electric vehicle charging. *IEEE Transactions on Power Systems*, 28(2):940–951, 2013.

[11] J. Gao, L. Sun, and M. Gen. A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. *Computers & Operations Research*, 35:2892–2907, 2008.

[12] J. García-Álvarez, M. González, and C. Vela. A genetic algorithm for scheduling electric vehicle charging. In *Proceedings of the Genetic and Evolutionay Computation Conference (GECCO 2015)*, pages 393–400, 2015.

[13] J. García-Villalobos, I. Zamora, J. San Martín, F. Asensio, and V. Aperribay. Plug-in electric vehicles in electric distribution networks: A review of smart charging approaches. *Renewable and Sustainable Energy Reviews*, 38:717–731, 2014.

[14] M. González and C. R. Vela. An efficient memetic algorithm for total weighted tardiness minimization in a single machine with setups. *Applied Soft Computing*, 37:506–518, 2015.

[15] I. Grau Unda, P. Papadopoulos, S. Skarvelis-Kazakos, L. Cipcigan, N. Jenkins, and E. Zabala. Management of electric vehicle battery charging in distribution networks with multi-agent systems. *Electric Power Systems Research*, 110:172–179, 2014.
[16] S. Hajforoosh, M. A. Masoum, and S. M. Islam. Real-time charging coordination of plug-in electric vehicles based on hybrid fuzzy discrete particle swarm optimization. *Electric Power Systems Research*, 128:19–29, 2015.
[17] P. Hansen, N. Mladenović, and J. Moreno Pérez. Variable neighborhood search: methods and applications. *Annals of Operations Research*, 175(1):367–407, 2010.
[18] Y. He, B. Venkatesh, and L. Guan. Optimal scheduling for charging and discharging of electric vehicles. *IEEE Transactions on Smart Grid*, 3(3):1095–1105, 2012.
[19] A. Hernandez-Arauzo, J. Puente, M. A. Gonzalez, R. Varela, and J. Sedano. Dynamic scheduling of electric vehicle charging under limited power and phase balance constraints. In *Proceedings of the 7th Scheduling and Planning Applications Workshop (SPARK 2013)*, pages 1–8, 2013.
[20] A. Hernandez-Arauzo, J. Puente, R. Varela, and J. Sedano. Electric vehicle charging under power and balance constraints as dynamic scheduling. *Computers & Industrial Engineering*, 85:306–315, 2015.
[21] M. Honarmand, A. Zakariazadeh, and S. Jadid. Optimal scheduling of electric vehicles in an intelligent parking lot considering vehicle-to-grid concept and battery condition. *Energy*, 65:572–579, 2014.
[22] X. Hu and B. Egardt. Advanced power-source integration in hybrid electric vehicles: Multicriteria optimization approach. *IEEE Transactions on Industrial Engineering*, 62(12):7847–7858, 2015.
[23] X. Hu, C. Zou, C. Zhang, and Y. Li. Technological developments in batteries: A survey of principal roles, types, and management needs. *IEEE Power and Energy Magazine*, 15(5):20–31, 2017.
[24] E. Iversen, J. Morales, and H. Madsen. Optimal charging of an electric vehicle using a markov decision process. *Applied Energy*, 123:1–12, 2014.
[25] C. Jin, J. Tang, and P. Ghosh. Optimizing electric vehicle charging: a customer's perspective. *IEEE Transactions on Vehicular Technology*, 62(7):2919–2927, 2013.
[26] C. Jin, J. Tang, and P. Ghosh. Optimizing electric vehicle charging with energy storage in the electricity market. *IEEE Transactions on Smart Grid*, 4(1):311–320, 2013.
[27] J. Kang, S. J. Duncan, and D. N. Mavris. Real-time scheduling techniques for electric vehicle charging in support of frequency regulation. *Procedia Computer Science*, 16:767–775, 2013.
[28] E. Karfopoulos and N. Hatziargyriou. A multi-agent system for controlled charging of a large population of electric vehicles. *IEEE Transactions on Power Systems*, 28(2):1196–1204, 2013.
[29] H. Kim, J. Lee, and G. Park. Constraint-based charging scheduler design for electric vehicles. In *Proceedings of the 4th Asian conference on Intelligent Information and Database Systems*, volume Part III, pages 266–275, 2012.
[30] J. Lopes, F. Soares, P. Almeida, and M. Moreira da Silva. Smart charging strategies for electric vehicles: Enhancing grid performance and maximizing the use of variable renewable energy resources. In *EVS24 International Battery, Hybrid and Fuel Cell Electric Vehicle Symposium*, pages 392–396, 2009.
[31] Z. Luo, Z. Hu, Y. Song, Z. Xu, and H. Lu. Optimal coordination of plug-in electric vehicles in power grids with cost-benefit analysis – part i: Enabling techniques. *IEEE Transactions on Power Systems*, 28(4):3546–3555, 2013.
[32] C. Ma, J. Rautiainen, D. Dahlhaus, A. Lakshman, J.-C. Toebermann, and M. Braun. Online optimal charging strategy for electric vehicles. *Energy Procedia*, 73:173–181, 2015.
[33] Z. Ma, D. Callaway, and I. Hiskens. Decentralized charging control of large populations of plug-in electric vehicles. *IEEE Transactions on Control Systems Technology*, 21(1):67–78, 2013.
[34] M. Masikos, K. Demestichas, E. Adamopoulou, and M. Theologou. Energy-efficient routing based on vehicular consumption predictions of a mesoscopic learning model. *Applied Soft Computing*, 28:114–124, 2015.
[35] M. J. Mirzaei, A. Kazemi, and O. Homaee. Real-world based approach for optimal management of electric vehicles in an intelligent parking lot considering simultaneous satisfaction of vehicle owners and parking operator. *Energy*, 76:345–356, 2014.
[36] N. Mladenović and P. Hansen. Variable neighborhood search. *Computers & Operations Research*, 24:1097–1100, 1997.
[37] N. Mladenović and P. Hansen. Variable neighborhood search: principles and applications. *European Journal of Operational Research*, 130:449–467, 2001.
[38] D. Oliveira, A. Zambroni de Souza, and L. Delboni. Optimal plug-in hybrid electric vehicles recharge in distribution power systems. *Electric Power Systems Research*, 98:77–85, 2013.
[39] I. Oliver, D. Smith, and J. Holland. A study of permutation crossover operators on the tsp. In *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 224–230, 1987.
[40] J. Puente, C. R. Vela, and I. González-Rodríguez. Fast local search for fuzzy job shop scheduling. In *Proceedings of ECAI 2010*, pages 739–744. IOS Press, 2010.
[41] I. Rahman, P. Vasant, B. Singh, M. Abdullah-Al-Wadud, and N. Adnan. Review of recent trends in optimization techniques for plug-in hybrid, and electric vehicle charging infrastructures. *Renewable and Sustainable Energy Reviews*, 58:1039–1047, 2016.
[42] I. Rahman, P. M. Vasant, B. S. M. Singh, and M. Abdullah-Al-Wadud. Swarm intelligence-based smart energy allocation strategy for charging stations of plug-in hybrid electric vehicles. *Mathematical Problems in Engineering*, 2015(Article ID 620425):10 pages, 2015.
[43] M. G. C. Resende and C. C. Ribeiro. Greedy randomized adaptive search procedures advances and applications. In M. Gendreau and J. Potvin, editors, *Handbook of Metaheuristics, 2nd edition*, chapter 10, pages 283–320. Springer, 2010.
[44] P. Richardson, D. Flynn, and A. Keane. Optimal charging of electric vehicles in low-voltage distribution systems. *IEEE Transactions on Power Systems*, 27(1):268–279, 2012.

[45] J. Sedano, M. Portal, A. Hernández Arauzo, J. Villar, J. Puente, and R. Varela. Sistema de control autónomo para distribución de energía en una estación de carga de vehículos eléctricos: diseño y operación. Technical report, Instituto Tecnológico de Castilla y León ITCL, 2012.

[46] J. Sedano, M. Portal, A. Hernández-Arauzo, J. R. Villar, J. Puente, and R. Varela. Intelligent system for electric vehicle charging: Design and operation. *DYNA*, 88(6):644–651, 2013.

[47] A. Sheikhi, S. Bahrami, A. Ranjbar, and H. Oraee. Strategic charging method for plugged in hybrid electric vehicles in smart grids; a game theoretic approach. *International Journal of Electrical Power & Energy Systems*, 53:499–506, 2013.

[48] S.-O. Shim and Y.-D. Kim. Scheduling on parallel identical machines to minimize total tardiness. *European Journal of Operational Research*, 177(1):135–146, 2007.

[49] F. Soares, P. Rocha Almeida, and J. Peças Lopes. Quasi-real-time management of electric vehicles charging. *Electric Power Systems Research*, 108:293–303, 2014.

[50] J. Soares, T. Sousa, H. Morais, Z. Vale, B. Canizes, and A. Silva. Application-specific modified particle swarm optimization for energy resource scheduling considering vehicle-to-grid. *Applied Soft Computing*, 13(11):4264–4280, 2013.

[51] W. Su and M.-Y. Chow. Computational intelligence-based energy management for a large-scale phev/pev enabled municipal parking deck. *Applied Energy*, 96:171–182, 2012.

[52] W. Su and M.-Y. Chow. Performance evaluation of an eda-based large-scale plug-in hybrid electric vehicle charging algorithm. *IEEE Transactions on Smart Grid*, 3(1):308–315, 2012.

[53] O. Sundstrom and C. Binding. Flexible charging optimization for electric vehicles considering distribution grid constraints. *IEEE Transactions on Smart Grid*, 3(1):26–37, 2012.

[54] T. Tran, M. Dogru, U. Ozen, and C. Beck. Scheduling a multi-cable electric vehicle charging facility. In *Proceedings of SPARK'13. ICAPS'13 Scheduling and Planning Applications woRKshop, ICAPS council*, pages 20–26, 2013.

[55] S. Vandael, B. Claessens, M. Hommelberg, T. Holvoet, and G. Deconinck. A scalable three-step approach for demand side management of plug-in hybrid vehicles. *IEEE Transactions on Smart Grid*, 4(2):720–728, 2013.

[56] C. R. Vela, R. Varela, and M. A. González. Local search and genetic algorithm for the job shop scheduling problem with sequence dependent setup times. *Journal of Heuristics*, 16:139–165, 2010.

[57] D. Wu, D. Aliprantis, and L. Ying. Load scheduling and dispatch for aggregators of plug-in electric vehicles. *IEEE Transactions on Smart Grid*, 3(1):368–376, 2012.

[58] X. Wu, X. Hu, S. Moura, X. Yin, and V. Pickert. Stochastic control of smart home energy management with plug-in electric vehicle battery energy storage and photovoltaic array. *Journal of Power Sources*, 333:203–212, 2016.

[59] S. Xu, D. Feng, Z. Yan, L. Zhang, N. Li, L. Jing, and J. Wang. Ant-based swarm algorithm for charging coordination of electric vehicles. *International Journal of Distributed Sensor Networks*, 2013(Article ID 268942):13 pages, 2013.

[60] J. Yang, L. He, and S. Fu. An improved PSO-based charging strategy of electric vehicles in electrical distribution grid. *Applied Energy*, 128:82–92, 2014.

[61] A. Zakariazadeh, S. Jadid, and P. Siano. Multi-objective scheduling of electric vehicles in smart distribution system. *Energy Conversion and Management*, 79:43–53, 2014.

[62] K. Zhan, Z. Hu, Y. Song, N. Lu, Z. Xu, and L. Jia. A probability transition matrix based decentralized electric vehicle charging method for load valley filling. *Electric Power Systems Research*, 125:1–7, 2015.

Figure 13: Jorge García-Álvarez received his M.Sc. in Computer Science in 2005 and is currently working in his Ph.D. degree in the University of Oviedo (Spain). He is currently an IT Consultant in Computer Science Corporation. His research interests include evolutionary computation and metaheuristic techniques applied to optimizations problems.

Figure 14: Miguel A. González received the M.Sc. in Computer Science in 2005 and the Ph.D. degree in 2011 in the University of Oviedo (Spain). He is currently a Lecturer in the Department of Computer Science of the University of Oviedo. His research interests include evolutionary computation techniques applied to optimization problems, particularly scheduling.

Figure 15: Camino R. Vela received the M.Sc. in Mathematics from the University of Zaragoza (Spain) in 1986 and the Ph.D. degree in Mathematics from the University of Oviedo (Spain) in 1990. She is currently Professor and Head of the Department of Computer Science, University of Oviedo. Her research interests include heuristic search, evolutionary computation, soft computing and other metaheuristic techniques and their application to constraint satisfaction problems such as scheduling. Dr. Vela is a member of the Spanish Association for Artificial Intelligence (AEPIA).

8.2. Genetic fuzzy schedules for charging electric vehicles

Presentamos la siguiente publicación:

- **Título:** Genetic fuzzy schedules for charging electric vehicles.
- **Revista:** *Computers & Industrial Engineering.*
- **Año:** 2018.
- **DOI:** 10.1016/j.cie.2018.05.019.
- **Referencia:** García-Álvarez, J.; González-Rodríguez, I.; Vela, C.R.; González, M.A.; Afsar, S. Genetic fuzzy schedules for charging electric vehicles. *Computers & Industrial Engineering.* 2018, 121, 51–61. DOI: 10.1016/j.cie.2018.05.019.

Author's copy of:
J. García-Álvarez, I. González-Rodríguez, C. R. Vela, M. A. González, S. Afsar.
Genetic fuzzy schedules for charging electric vehicles.
Computers & Industrial Engineering 121 (2018) 51-61.
DOI of published version: 10.1016/j.cie.2018.05.019

Genetic Fuzzy Schedules for Charging Electric Vehicles

Jorge García-Álvarez[a], Inés González-Rodríguez[b], Camino R. Vela[a], Miguel A. González[a,*], Sezin Afsar[c]

[a] *Dept. of Computing, University of Oviedo,(Spain)*
[b] *Dept. of Mathematics, Statistics and Computing, University of Cantabria, (Spain)*
[c] *Independent researcher*

Abstract

This work tackles the problem of scheduling the charging of electric vehicles in a real-world charging station subject to a set of physical constraints, with the goal of minimising the total tardiness with respect to a desired departure date given for each vehicle. We model a variant of the problem that incorporates uncertainty in the charging times using fuzzy numbers. As solving method, we propose a genetic algorithm with tailor-made operators, in particular, a new chromosome evaluation method based on generating schedules from a priority vector. Finally, an experimental study avails the proposed genetic algorithm both in terms of algorithm convergence and quality of the obtained solutions.

Keywords: electric vehicle, charging station, scheduling, genetic algorithm, fuzzy number, heuristic

1. Introduction

Perceived as futuristic prototypes not so long ago, electric vehicles (EVs) and all the technology surrounding them have experimented an extraordinary growth in the last years, to the point that now EVs are seen as a real alternative to fossil-fuelled vehicles, producing an increasing impact on the economy and the environment. Clearly, they reduce the dependency on petrol and promote alternative more environmentally-friendly sources of energy. Also, they can act as valuable distributed energy resources, smoothing intermittency due to renewable energy sources and supporting grid-wide frequency stability (Kang et al., 2013).

The use of EVs also presents several technological challenges, as deciding the optimal locations of new vehicle charging stations You and Hsieh (2014) or the development of smart systems that manage charging grids in order to

*Corresponding author
Email addresses: jgarcia-alvarez@outlook.com (Jorge García-Álvarez), gonzalezri@unican.es (Inés González-Rodríguez), crvela@uniovi.es (Camino R. Vela), mig@uniovi.es (Miguel A. González), sezinafsar@gmail.com (Sezin Afsar)

optimally distribute and balance electricity consumption while satisfying EVs' electricity demands. A great variety of problems arise depending, for instance, on the charging infrastructure or the objectives to be satisfied (Rahman et al., 2016). Recent reviews of methods and strategies to manage electricity grids and schedule EVs charging can be found in Hernandez-Arauzo et al. (2015).

Here we consider a problem that consists in scheduling the charging of a set of EVs taking into account the characteristics of a real-life charging station described in Sedano et al. (2013). The station has been designed to be installed in private car parks under simplicity and economy criteria. Each parking place in the car park is owned by a particular user and has a charging point. Every charging point is connected to one of the lines of a three-phase electric feeder, with a centralised control that establishes the power available to the point at any time. There are power constraints limiting the number of EVs that can be simultaneously charging on one line. There is also a balance constraint that limits the difference in active charging points between different lines. It is this balance constraint that poses the most relevant difference with respect to other charging models in the literature.

We will tackle a variant of the static scheduling problem for the charging station presented by Hernandez-Arauzo et al. (2015). Their problem statement assumes that the set of EVs that need to be charged within a planning horizon is known in advance, together with the exact arrival time, desired departure time and charging time of each EV. The goal is to provide a charging schedule for all the EVs that is feasible in the sense that all technical constraints hold and such that the total tardiness with respect to the desired departure times is minimised. This static version of the problem is of great importance, since it constitutes the basis for solving a dynamic and hence more realistic version. However, it assumes that charging times are exactly known in advance, an assumption that might be deemed as unrealistic. We thus propose in this work to narrow the gap between the academic model and the real situation by incorporating uncertainty into charging times.

We propose to model uncertain charging times using fuzzy sets. The use of fuzzy sets to model uncertain durations in scheduling is well established. Among others, Chanas and Kasperski (2001) minimise maximum lateness in a single machine scheduling problem with fuzzy processing times using an extension of Lawler's algorithm, and the parallel machine scheduling problem with fuzzy processing times is tackled in Balin (2011). We also find numerous works focused on shop scheduling problems with fuzzy durations and, in particular, in the job shop in its multiple variants Liu et al. (2015) Abdullah and Abdolrazzagh-Nezhad (2014). Fuzzy random theory have been successfully used to model imprecise parameters in real-life problems, such as a location-allocation problem in a container freight station Zhong et al. (2015).

There are a few examples where fuzzy sets have also been used in problems related with EV charging. However, they are mostly used to model preferences or soft constraints and to perform multicriteria decision making, for instance, to decide on the optimal site selection of EV charging stations in Guo and Zhao (2015) or to coordinate EV charging in an electric grid in Hajforoosh et al.

(2015). More related to our work, in Ansari et al. (2015) fuzzy sets model electricity market uncertainties with the aim of optimising the coordinated bidding of EVs used as ancillary services in the electricity grid. To our knowledge, this is the only proposal in the literature to use fuzzy sets to model uncertain parameters in problems related to EVs, making ours the first attempt to schedule the charging of EV considering fuzzy times.

The remaining of this paper is organised as follows. Section 2 will provide the necessary background on fuzzy numbers and a new result to handle fuzzy charging times. Then, in Section 3 we will show how the use of triangular fuzzy numbers to model charging times leads to a new formulation of the problem. In Section 4 we will propose a first solving method for the resulting fuzzy problem. This is a genetic algorithm with indirect encoding, based on a competitive method for the deterministic problem given in García-Álvarez et al. (2015), but where some operators need to be adapted to the fuzzy setting. In particular, the decoding operator will consist in a new purpose-built algorithm to obtain feasible charging schedules from a priority vector. The resulting method will be evaluated with an experimental study in Section 5. Finally, some conclusions and perspectives for future work will be given in Section 6.

2. Uncertainty in charging times

In real-life problems it is frequent that different sources of uncertainty exist and some input data are not precisely known; in our case, the time it will take for an EV to charge is not exactly known in advance. Currently, crisp charging times are estimated from available historical data, despite the low quality of these data. The alternative of modelling these ill-known times instead with probability distributions is not an option, since it would require to have more and better data. Also, the resulting problem would be excessively complex from the computational point of view. In cases such as this, fuzzy sets constitute a very interesting alternative that may help building a tradeoff between the expressive power and the computational difficulties of stochastic scheduling techniques while tackling uncertainty (Dubois et al., 2003, 2008; Wong and Lai, 2011).

A fuzzy quantity $\widetilde{q}$ is a fuzzy set on the reals $\mathbb{R}$ with membership function $\mu_{\widetilde{q}} : \mathbb{R} \to [0, 1]$. The *$\alpha$-cuts* of a fuzzy quantity are given by $\widetilde{q}_\alpha = \{r \in \mathbb{R} : \mu_{\widetilde{q}}(r) \geq \alpha\}, \alpha \in (0, 1]$, and its *support* is defined as $\widetilde{q}_0 = \{r \in \mathbb{R} : \mu_{\widetilde{q}}(r) > 0\}$. A *fuzzy interval* is a fuzzy quantity whose α-cuts are intervals (bounded or not) and a *fuzzy number* $\widetilde{m}$ is a fuzzy quantity with compact support and unique modal value whose α-cuts are closed intervals, denoted $\widetilde{m}_\alpha = [\underline{m}_\alpha, \overline{m}_\alpha]$. In our context, a fuzzy interval is seen as a possibility distribution, representing more or less likely possible values for a charging time.

Here we propose that each uncertain charging time be modelled using a *triangular fuzzy number* or TFN, given by an interval $[a^1, a^3]$ of possible values (its support) and a unique modal value $a^2 \in [a^1, a^3]$ with $\mu_{\widehat{a}}(a^2) = 1$. Hence, a TFN $\widehat{a}$ can be denoted $\widehat{a} = (a^1, a^2, a^3)$ and its membership value is given by

the following equation:

$$\mu_{\widehat{a}}(x) = \begin{cases} \frac{x-a^1}{a^2-a^1} & : a^1 \leq x \leq a^2 \\ \frac{x-a^3}{a^2-a^3} & : a^2 < x \leq a^3 \\ 0 & : x < a^1 \text{ or } a^3 < x \end{cases} \tag{1}$$

Notice that any real number $a \in \mathbb{R}$ can be seen as a particular case of TFN $\widehat{a} = (a, a, a)$ with all its defining points equal to a.

2.1. Arithmetic of TFNs

To handle charging times given as TFNs, we need to be able to compute the addition, substraction and maximum of two TFNs. In principle, the arithmetic operations on TFNs can be obtained by extending the corresponding operations on real numbers using the *Extension Principle* Dubois and Prade (1986). Both the addition and the substraction of two TFNs $\widehat{a}$ and $\widehat{b}$ are TFNs given by

$$\widehat{a} + \widehat{b} = (a^1 + b^1, a^2 + b^2, a^3 + b^3) \tag{2}$$

$$\widehat{a} - \widehat{b} = (a^1 - b^3, a^2 - b^2, a^3 - b^1) \tag{3}$$

Regarding the maximum, computing the resulting expression is cumbersome, if not intractable; also, the set of TFNs is not closed under this operation. For the sake of simplicity and tractability of numerical calculations, it is fairly common in the literature to approximate the maximum, either using a ranking method, as it is done in Lei (2010), or by interpolation, following Fortemps (1997), evaluating only the operation on the three defining points of each TFN, that is:

$$\max(\widehat{a}, \widehat{b}) \approx \max_I(\widehat{a}, \widehat{b}) = (\max(a^1, b^1), \max(a^2, b^2), \max(a^3, b^3)). \tag{4}$$

The approximation max_I has been widely used in the scheduling literature, from earlier works such as (Fortemps, 1997; Kuroda and Wang, 1996) to more recent ones (Palacios et al., 2015a; Wang et al., 2010), to mention but a few. Additional arguments to support this approximation can be found in (Palacios et al., 2017).

Unless otherwise stated and for the sake of a simpler notation, we shall simply write max when referring to the interpolated maximum $\max_I$.

2.2. Expected value

When working with fuzzy numbers, it is often useful in practice to obtain their expected value, similarly to what is done with probability distributions in stochastic settings. In particular, for a TFN $\widehat{a}$, its expected value is given by:

$$E[\widehat{a}] = \frac{1}{4}(a^1 + 2a^2 + a^3). \tag{5}$$

This expression is obtained following different approaches, among others as the expected value of a fuzzy number based on random sets (Heilpern, 1992), as the

generative expected value induced by the evidence $\widehat{a}$ (Chanas and Nowakowski, 1988), as the centre of the mean value of $\widehat{a}$ (Dubois and Prade, 1987) and as the expected value of the so-called pignistic probability distribution that is the centroid of the set of probabilities $\mathcal{P}(\Pi_{\widehat{a}})$ dominated by the possibility measure associated with $\widehat{a}$ (Dubois, 2006).

Notice that, for a TFN $\widehat{a}$ it is always the case that the expected value lies in its support, that is, $a^1 \leq E[\widehat{a}] \leq a^3$. It is also easy to provide an alternative expression for the expected value as follows:

$$E[\widehat{a}] = a^2 + \frac{(a^3 - a^2) - (a^2 - a^1)}{4}. \tag{6}$$

An added value of expected values is that they allow to compare fuzzy numbers. Indeed, no natural order exists in the set of fuzzy numbers, so numerous ranking methods have been and keep being proposed in the literature (Brunelli and Mezei, 2013). Several of these methods have been used in the field of fuzzy scheduling (cf. (Abdullah and Abdolrazzagh-Nezhad, 2014; Dubois et al., 2003)). In particular, the expected value defines an index-based ranking method, inducing a total ordering $\leq_E$ in the set of TFNs (Fortemps, 1997), where for any two TFNs $\widehat{a}, \widehat{b}$,

$$\widehat{a} \leq_E \widehat{b} \text{ if and only if } E[\widehat{a}] \leq E[\widehat{b}]. \tag{7}$$

Notice that $\leq_E$ coincides with several other ranking methods from the literature which are not based on expected values, as highlighted in (Palacios et al., 2015b). Additionally, Brunelli and Mezei (2013) present a numerical study that suggests that, for TFNs, the ranking based on the expected value is very similar to seven more ranking methods, in the sense that the ordering they induce in a sample of TFNs is strongly correlated.

Interestingly, it is also possible to establish a relationship between the ranking method based on expected values and classical interval comparison in the light of imprecise probabilities (Destercke and Couso, 2015; Dubois, 2011). In particular, it comes down to using Hurwicz criterion for classical interval comparison on upper and lower expectations derived from A. This provides us with an interpretation for comparisons based on $\leq_E$ as those corresponding to a decision maker who keeps an equilibrium between pessimism and optimism.

2.3. Defining a TFN subject to a set of constraints

When scheduling the charge of EVs, we will need to determine fuzzy starting times for the charging of each EV subject to a set of constraints. The following result gives us a means to do so.

Theorem 1. *Let $e \in \mathbb{R}$ be a number and $\widehat{c} = (c^1, c^2, c^3)$ a TFN such that $E[\widehat{c}] \leq e$. A TFN $\widehat{s} = (s^1, s^2, s^3)$ satisfying:*

$$E[\widehat{s}] = e \tag{R_0}$$

$$c^i \leq s^i, i = 1, 2, 3 \tag{$R_{1,2,3}$}$$

is given by the following expression:

$$\begin{aligned}
&\textit{if } E[\widehat{c}] = e, && \widehat{s} = \widehat{c};\\
&\textit{if } E[\widehat{c}] < e, c^3 - e \leq e - c^1, && \widehat{s} = (s^1, s^2, s^3) \textit{ where } \begin{cases} s^3 &= \max(e, c^3),\\ s^2 &= \max(e, c^2),\\ s^1 &= 4e - 2s^2 - s^3; \end{cases}\\
&\textit{if } E[\widehat{c}] < e, c^3 - e > e - c^1, && \widehat{s} = (s^1, s^2, s^3) \textit{ where } \begin{cases} s^1 &= c^1,\\ s^3 &= c^3,\\ s^2 &= \frac{1}{2}(4e - s^1 - s^3). \end{cases}
\end{aligned} \tag{8}$$

Proof. The definition of $\widehat{s}$ in (8) implies that R_0 always holds. Let us now prove that $s^i \geq c^i$, $i = 1, 2, 3$. We distinguish three cases depending on how $\widehat{s}$ is defined in (8):

1. If $E[\widehat{c}] = e$, it follows immediately since $\widehat{s} = \widehat{c}$.
2. If $E[\widehat{c}] < e$ and $c^3 - e \leq e - c^1$, then, $s^3 = \max(e, c^3) \geq c^3$ and $s^2 = \max(e, c^2) \geq c^2$. If $e > c^3$, it turns out that $s^3 = s^2 = s^1 = e > c^1$. If $e \leq c^3$ and therefore $0 \leq c^3 - e \leq e - c^1$, we prove that $s^1 \geq c^1$ by distinguishing 2 subcases:
   (a) If $c^2 < e$, $s^2 = e$ and $s^3 = c^3$, so (8) together with the fact that $c^3 - e \leq e - c^1$ mean that:
   $$s^1 = 2e - c^3 = e - (c^3 - e) \geq e - (e - c^1) = c^1.$$
   (b) If $e \leq c^2$, $s^3 = c^3$ and $s^2 = c^2$. But since $e = E[\widehat{s}]$,
   $$E[\widehat{s}] = c^2 + \frac{(c^3 - c^2) - (c^2 - s^1)}{4} = e \leq c^2,$$
   if and only if
   $$(c^3 - c^2) - (c^2 - s^1) \leq 0,$$
   or equivalently,
   $$s^1 \geq c^2 + c^2 - c^3.$$
   But also $E[\widehat{c}] < e$, that is:
   $$E[\widehat{c}] = c^2 + \frac{(c^3 - c^2) - (c^2 - c^1)}{4} < e \leq c^2,$$
   and therefore, $c^3 - c^2 < c^2 - c^1$, that is, $c^2 - c^3 > c^1 - c^2$. In consequence,
   $$s^1 \geq c^2 + c^2 - c^3 > c^2 + c^1 - c^2 = c^1.$$

3. If $E[\widehat{c}] < e$ and $c^3 - e > e - c^1$, then $s^1 = c^1$ y $s^3 = c^3$. Let us now show that $s^2 \geq c^2$. Indeed, $E[\widehat{c}] < e$ if and only if:

$$c^2 < \frac{4e - c^1 - c^3}{2}.$$

but since $c^1 = s^1$, $c^3 = s^3$, by definition of $\widehat{s}$:

$$s^2 = \frac{4e - c^1 - c^3}{2},$$

hence $c^2 < s^2$.

3. Problem Description

As mentioned earlier, our problem is motivated by a real charging station, described in detail by Sedano et al. (2013). The station was designed to be installed in a community car park consisting of privately-owned parking spaces . Its general scheme is given in Figure 1. The charging grid is fed by a three-phase electric power source, resulting in three so-called charging lines. Each parking space has a charging point connected to one of the lines but the contracted power limits the number of charging points that can be simultaneously fed on each line. Additionally, due to legal and technical reasons, there must be a load balance between different lines, so the power consumption (i.e. number of active charging points) does not differ much from one line to another. The station is controlled by a distributed system with a master agent per line controlling a number of slaves. Each slave in turn controls two consecutive charging points on the same line and is responsible for activating and deactivating them, as well as recording asynchronous events such as the arrival of a new vehicle. The master gathers information from the slaves and sends connection and disconnection orders to them, after accessing the database where the vehicles' data and the charging schedule are stored.

When users park their vehicles in their own parking spaces they need to provide a desired departure time. These values are used by the intelligent system to schedule the charging of all vehicles within a time frame. As proposed in Hernandez-Arauzo et al. (2015), we assume a simplified model of the problem whereby a user never departs before the vehicle has finished charging, so the objective is to minimise the tardiness of the charge completion time with respect to the desired departure date given by the user.

3.1. Problem formulation

Following the general problem description, the Electric Vehicles Charging Scheduling Problem with Fuzzy charging times or FuzEVCSP in short is formulated as follows.

In an instance of the FuzEVCSP problem there are three charging lines $L_i, 1 \leq i \leq 3$. Each line L_i has n_i charging points and there is a maximum

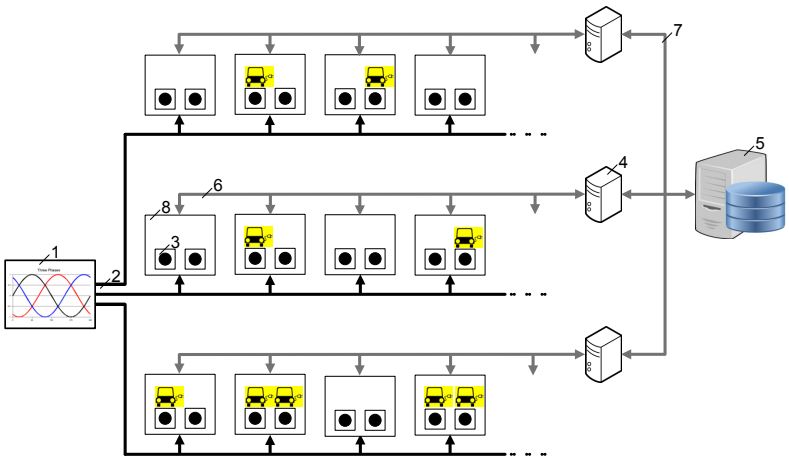


Figure 1: General structure of the charging station network and its components: (1) power source, (2) three-phase electric power, (3) charging points, (4) masters, (5) server with database, (6) communication RS 485, (7) communication TCP/IP, and (8) slaves.

number $N > 0$ of charging points that can be simultaneously active. Line L_i receives M_i electric vehicles $v_{i1}, \ldots, v_{iM_i}$ between time 0 and a planning horizon H. Each vehicle v_{ij} has an arrival time $t_{ij} \geq 0$, an uncertain charging time $\widehat{p_{ij}} > 0$ (represented as a TFN) and a departure time d_{ij} by which the vehicle should ideally be fully charged, representing the time when the driver wishes to remove the vehicle. Here we assume that the arrival and departure times are crisp times the user. Additionally, we assume that they are consistent with charging times in the sense that $d_{ij} \geq t_{ij} + p_{ij}^3$. The rationale here is that the departure time should allow for the vehicle to be fully charged provided that it starts charging as soon as it arrives, even if it takes the longest possible charging time.

A *feasible schedule* is an assignment of charging starting times $\widehat{st_{ij}}$ to all EVs $v_{ij}, 1 \leq i \leq 3, 1 \leq j \leq M_i$ such that the following constraints hold.

3.1.1. Constraints

Arrival time. All vehicles must start charging after their arrival time:

$$\forall v_{ij}, \quad st_{ij}^1 \geq t_{ij} \tag{9}$$

No preemption. Once a vehicle starts charging, it cannot be disconnected before it finishes charging. That is, if $\widehat{c_{ij}}$ denotes the time when v_{ij} finishes charging, it must be the case that:

$$\forall v_{ij}, \quad \widehat{c_{ij}} = \widehat{st_{ij}} + \widehat{p_{ij}} \tag{10}$$

Maximum active charging points per line. The number of charging points that are active on a line at any instant cannot exceed a given threshold N. define the set of EVs that may be charging on the same line when v_{ij} starts charging as:

$$A_{ij} = \{v_{ik} : \exists h \in \{1,2,3\} \; {}^{h}_{ik} \leq^{h}_{ij}, \max(,) \neq\}, \tag{11}$$

and $|A_{ij}|$ denotes its cardinal, then the constraint can be expressed as follows:

$$\forall v_{ij}, \quad |A_{ij}| \leq N. \tag{12}$$

Balance between lines. The load of active charging points must be evenly distributed among lines. Originally, for the deterministic problem, if $N_i(t)$ denotes the number of active charging points on line L_i, Hernandez-Arauzo et al. (2015) modelled this constraint using a threshold $\Delta \in [0,1]$ representing the maximum imbalance between any two different lines L_i and L_j, so that:

$$\forall t \geq 0, \; \max_{1 \leq i,j \leq 3} \left(\frac{|N_i(t) - N_j(t)|}{N} \right) \leq \Delta. \tag{13}$$

In a deterministic setting, we consider that a vehicle v_{ij} is charging at instant t if $st_{ij} \leq t \leq c_{ij}$. We extend this concept to the fuzzy setting by taking t as a TFN $\widehat{t} = (t,t,t)$, so the number of (expected) active charging points at instant $t \geq 0$ on line L_i, $1 \leq i \leq 3$, is defined as:

$$N_i(t) = \sum_{j=1}^{M_i} \delta_{ij}(t) \tag{14}$$

where

$$\delta_{ij}(t) = \begin{cases} 1 & : \text{ if } \widehat{st_{ij}} \leq_E \widehat{t} \leq_E \widehat{c_{ij}} \\ 0 & : \text{ otherwise.} \end{cases} \tag{15}$$

Thus, this constraint refers now to the expected load balance (since $N_i(t)$ is defined in terms of expected values).

3.1.2. Goal

The *goal* is to find a feasible solution minimising the total tardiness with respect to departure dates, defined as follows:

$$\widehat{T} = \sum_{i=1}^{3} \sum_{j=1}^{M_i} \max(0, \widehat{c_{ij}} - d_{ij}) \tag{16}$$

Since the objective value $\widehat{T}$ is a TFN, solutions will be compared using $\leq_E$.

4. Genetic Algorithm

Genetic algorithms or GAs in short are metaheuristic methods capable of efficiently exploring large search spaces. They also allow to incorporate heuristic

knowledge on the problem domain. Thus, they constitute solving methods for scheduling problems that yield good quality solutions with reasonable computational effort (Talbi, 2009).

Here, we propose a GA based on that proposed by García-Álvarez et al. (2015) for the deterministic version of the EV charging problem. This GA starts from an initial population composed by a mixture of random and heuristic individuals. It then evolves the initial population applying genetic operators until a stopping criterion is met, more precisely, until a solution with null total tardiness $\widehat{T} = 0$ is found or after a given number max_{iter} of consecutive iterations have passed without improving the fitness of the best individual in the population. The pseudocode for this GA can be seen in Algorithm 1.

The GA uses an indirect encoding, so in order to build a solution it makes use of a purpose-built schedule generation scheme that schedules the EV charging according to the information coded in the chromosome. This decoding operator has been newly designed for the problem at hand, while the remaining operators have been adapted from the existing GA, taking into account the differences between the deterministic and the fuzzy setting, as explained in the following subsections.

Input A problem instance P
Output A schedule S for P
  Generate the initial population;
  while Termination criterion is not satisfied **do**
    Group all chromosomes randomly in pairs;
    Apply crossover operator to each pair with probability p_X;
    Apply mutation operator to every offspring with probability p_M;
    Evaluate new chromosomes;
    Apply replacement strategy to pass chromosomes onto the next generation;
  end while
  return The schedule S from the best chromosome;

Algorithm 1: The genetic algorithm

4.1. Chromosome representation and evaluation

Chromosomes code solutions as a permutation of the set of EVs. To build a charging schedule S from a permutation V, we need a schedule-generation algorithm that assigns charging starting times to all EVs based on the priority vector represented by V. To this end, we propose here a schedule generation scheme that takes into account the constraints and characteristics of the fuzzy problem. The rationale is to sequentially assign to each EV the earliest possible starting time such that the problem constraints as described in Section 3.1 hold with respect to the already existing partial schedule.

The detailed procedure can be seen in Algorithm 2, where $\text{fuzz}(\widehat{c}, e)$ denotes the TFN built from $\widehat{c}$ and e according to Theorem 1. This scheme divides every

Input A permutation V of the EVs
Output A schedule S of charging starting times for all EVs
Consider every line L_i divided into N sublines $l_i^k : k = 1 \dots N$;
Initialise completion charging time on each subline $\leftarrow 0$
for all vehicle $v_{ij} \in V$ **do**
//Select subline where v_{ij} can start charging the earliest
$\widehat{est}^k \leftarrow \max(, t_{ij}),\ k = 1, \dots, N$
$k^* \leftarrow \arg\min\{E[\widehat{est}^k]\}$
//Find interval from st_{ij}^E with balance between lines
$st_{ij}^E \leftarrow \min\{t \geq E[\widehat{est}^{k^*}] : (13)$ holds in the interval $[t, t + \widehat{p_{ij}})\}$
//Move to the subline where keeping balance generates least idleness
$k^* \leftarrow \arg\max\{E[\widehat{est}^k] : E[\widehat{est}^k] \leq st_{ij}^E\}$
//Assign charging starting time and update subline completion times
$\widehat{st_{ij}} \leftarrow \text{fuzz}(\widehat{est}^{k^*}, st_{ij}^E)$
$\widehat{c_i}^{k^*} \leftarrow \widehat{st_{ij}} + \widehat{p_{ij}}$
end for
return The schedule S

Algorithm 2: Schedule generation scheme

line L_i into N virtual sublines l_i^k, $1 \leq k \leq N$, where N is the maximum number of active charging points allowed per line. By doing so, the constraint in (12) regarding the maximum number of active charging points holds immediately. Then, charging starting times for all vehicles are assigned in the order in which they appear in the input vector V. For a particular EV v_{ij}, its earliest possible starting time on each subline l_i^k, denoted $\widehat{est}^k$, is computed as the maximum between its arrival time t_{ij} and the time $\widehat{c_i}^k$ when the last EV already scheduled on that subline finishes charging. This guarantees that the first constraint in (9) holds. The line with the smallest earliest possible starting time (according to $\leq_E$) is selected as a candidate to schedule v_{ij} on that subline. However, before a starting time can be assigned to v_{ij}, we need to ensure that the balance between lines, as expressed in (13), holds with respect to the already scheduled EVs for all the charging time . This might imply delaying the start of charging of v_{ij} so its expected starting time coincides with an instant st_{ij}^E posterior to the expected value of its earliest possible starting time, with a big gap in between. The schedule building scheme thus tries to find another subline where keeping the balance generates the least idleness. It is only after this step that the subline k^* where v_{ij} should be charged is selected. The starting time is then established as the fuzzy time that can be obtained from the earliest possible starting time $\widehat{est}^{k^*}$ on that subline and the earliest instant where balance holds st_{ij}^E, according to Theorem 1. Notice that, according to that result, $E[] = st_{ij}^E$ (so balance holds) and $= \max(, \widehat{est}^{k^*})$ (so the charging time is posterior to the

EV's arrival and the constraint regarding maximum number of active charging points holds as well).

Let us illustrate this schedule-generation procedure with a toy problem where a total of 6 EVs must be scheduled: four EVs $\{v_{11}, v_{12}, v_{13}, v_{14}\}$ on line L_1, one EV $\{v_{21}\}$ on L_2 and one EV $\{v_{31}\}$ on L_3. The arrival time is 0 for every EV except for v_{14}, with $t_{14} = 8$. The charging times are given by $\widehat{p_{12}} = \widehat{p_{21}} = \widehat{p_{31}} = (4, 5, 6)$, $\widehat{p_{11}} = (10, 11, 12)$, $\widehat{p_{13}} = (14, 15, 17)$, and $\widehat{p_{14}} = (5, 6, 7)$, and departure times are $d_{12} = d_{21} = d_{31} = 9$, $d_{11} = 11$, and $d_{13} = d_{14} = 20$. The maximum number of active charging points on every line is $N = 3$ and the maximum imbalance threshold is $\Delta = \frac{2}{3}$, so at every instant there should be at most an expected load difference of 2 EVs between any two lines.

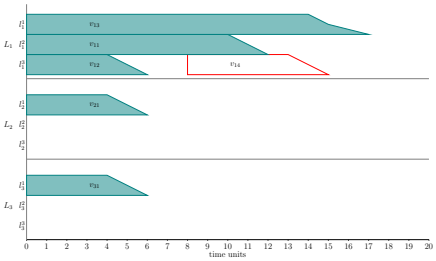
Figure 2 illustrates the scheduling procedure for the situation where the algorithm's input is the permutation $(v_{11}, v_{21}, v_{31}, v_{12}, v_{13}, v_{14})$ and all EVs have been scheduled except for the last one v_{14}. In order to schedule it, we search for the subline k in L_1 where v_{14} could start to charge the earliest, which turns out to be l_1^3, with $\widehat{est}^3 = (8, 8, 8)$ (Figure 2(a)). Then, we compute the first instant $st_{ij}^E \geq 8$ after which balance between lines holds for the whole charging time of v_{14}, yielding $st_{ij}^E = 11$ (Figure 2(b)). We then notice that v_{14} can be scheduled from that instant on a different subline l_1^2 (Figure 2(c)). Hence, the value for $\widehat{st_{14}}$ is obtained from $st_{14}^E = 11$ in such a way that the EV can start charging after the earliest starting time on that subline $\widehat{est^2}$, that is, $\widehat{st_{14}} = (10, 11, 12)$.

4.2. Genetic operators

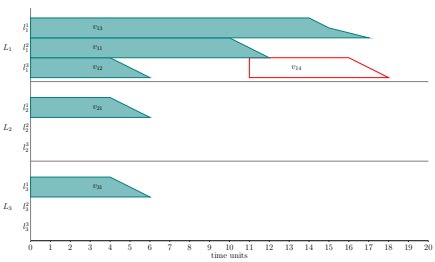
Except for the above decoding algorithm, the rest of the GA follows closely the one proposed by García-Álvarez et al. (2015) for the deterministic version of the problem. Let us briefly describe its main characteristics.

To generate an *initial population* both diverse and containing good individuals, one third of the individuals are generated as random permutations and each one of the other two thirds is generated by applying a stochastic version of a different dispatching rule. The first rule, DDR (Due-Date Rule) orders EVs in ascending order according to their departure time d_{ij}. The second rule, EST (Earliest Starting Time) orders EVs in ascending order according to their arrival time t_{ij}. The stochastic versions of both rules select the next EV to be added to the chromosome by applying tournament to *tourn* randomly selected individuals, where *tourn* is a parameter of the algorithm.

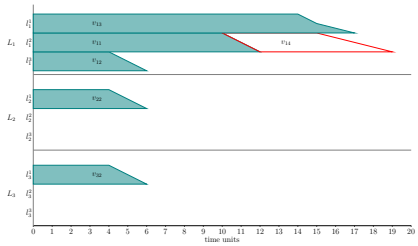
In the *selection phase*, all chromosomes are randomly paired. Then, crossover is applied (with probability p_X) to each pair in order to obtain two offspring, which may be subject to mutation with probability p_M. The *replacement strategy* selects from each set of parents and offspring the best two individuals with different fitness (i.e. expected tardiness) values. The goal is to maintain diversity in the population. If both parents and both offspring all have the same tardiness, then two of them are selected at random. Notice that this replacement strategy has an elitism effect, since the best individual in a generation is always equal to or better than the best individual in the previous generation.



(a) Arrival of v_{14} to line L_1 at $t_{14} = 8$. The EV could be scheduled on the third subline l_1^3 with starting time $\widehat{8}$ if no balance was required.



(b) Balance holds only after $st_{14}^E = 11$



(c) v_{14} is finally scheduled on l_1^2 with $\widehat{st_{14}} = (10, 11, 12)$

Figure 2: Example of schedule generation

Both crossover and mutation operators are those from (García-Álvarez et al. 2015), adapted to handling fuzzy times. The *crossover operator* SBX (Starting-time Based Crossover) was designed so offspring are likely to improve the expected tardiness value of their parents. The operator randomly selects an instant t_0, so the first offspring is generated by selecting from the first parent all EVs that are expected to start charging before t_0, while the remaining ones are rearranged according to the second parent. The second offspring is created similarly, with the first and second parent exchanging roles, that is, those EVs expected to start earlier than t_0 in the second parent pass onto the offspring in the same relative order and the remaining EVs are re-ordered following the first parent.

Operator SBX is illustrated in Figure 3. Provided that the randomly selected instant is $t_0 = 9$, to generate the first offspring O_1 we select from P_1 all EVs that are expected to start charging before t_0, that is, those v_{ij} such that $E[\widehat{s}_{ij}] < t_0$: $3, 7, 1, 8, 2, 4, 10$ and 12. These EVs are inserted in O_1 preserving their relative order. The chromosome is later completed with the remaining EVs (those expected to start after t_0 in P_1) in the same relative order as they appear in P_2, that is, $6, 5, 11, 9$. The second offspring is obtained similarly.

The *mutation operator* mutates each line L_i with probability p_M. If line L_i is mutated, a random subset of consecutive EVs is selected on that line and this set is then shuffled, re-ordering these EVs. Figure 4 shows an example of

Parents

$\mathbf{P_1}$	3	7	1	8	2	4	6	5	10	9	12	11
$\mathbf{P_2}$	8	6	5	2	11	9	7	3	10	12	1	4

Expected charge starting times

$\mathbf{E[S]_1}$	0	0	0	0	0	7	10	10	0	10	7	11
$\mathbf{E[S]_2}$	0	10	0	0	7	0	0	5	9	12	0	7

Offspring

$\mathbf{O_1}$	3	7	1	8	2	4	10	12	**6**	**5**	**11**	**9**
$\mathbf{O_2}$	8	5	2	11	9	7	3	1	4	**6**	**10**	**12**

Figure 3: Example of crossover SBX. We highlight in red those vehicles with expected charge starting time (also in red below) greater than or equal to $t_0 = 9$.

mutation, supposing that only line L_2 is to be mutated. The random subset of consecutive EVs on that line that is selected is $2, 6, 12$. These are shuffled and reinserted in the original chromosome in the same positions occupied by this subset, but with the new relative order.

Initial chromosome

$\mathbf{O_1}$	3	7	1	8	**2**	4	**6**	5	10	9	**12**	11

	Matrix representation						*Selected vehicles*				*Shuffled vehicles*		
L_1	3	1	4	5									
L_2	8	2	6	12	11	⇒	2	6	12	⇒	6	12	2
L_3	7	10	9										

Mutated chromosome

$\mathbf{O_1}$	3	7	1	8	**6**	4	**12**	5	10	9	**2**	11

Figure 4: Example of mutation, where a random subset of consecutive EVs on line 2 (highlighted in red) is shuffled.

5. Experimental results

We now proceed to empirically evaluate the proposed method, which has been implemented in C++ and run on a PC with a Xeon E5520 at 2.2Ghz processor and 24Gb RAM running Linux (SL 6.0).

As benchmark instances, we take fuzzified versions of the problem instances used in Hernandez-Arauzo et al. (2015), which are based on real data. We simulate uncertainty in charging times using the fuzzification method from Ghrayeb (2003). In all cases, the charging station is supposed to be installed in a car park with 180 parking spaces. The planning horizon is 1 day and there are different profiles of arrival, charging and departure times based on real data. Instances are divided in two groups: . Clearly, situations of imbalance between lines are more likely to arise on instances of *Type 2*, something that the algorithm must

take into account to build feasible schedules. Finally, we also consider different values for the imbalance parameter $\Delta \in \{0.2, 0.4, 0.6, 0.8\}$ and for the maximum charging points on a line $N \in \{20, 30, 40\}$. In total there are 24 instances, one per combination of *Type*, Δ and N values.

After an initial parametric study, the algorithm parameters have been fixed as follows: population size is 200 individuals, the stopping parameter of maximum number of non-improving iterations is set to $max_{iter} = 25$, crossover and mutation probabilities are $p_X = 0.8$ and $p_M = 0.1$ respectively and tournament size for the heuristic initial population is set to $tourn = 8$. All experiments correspond to 30 runs of the GA on each instance, in order to obtain statistically significant results.

5.1. Algorithm's convergence

First, we analyse the GA's evolution. To do so, we compare its results with a heuristic algorithm without evolution that generates as many groups of 200 individuals as generations takes for the GA to converge. Each group is formed by random individuals and individuals generated with the stochastic versions of rules DDR and EST in the same proportion as in the initial population of the GA. In order to make comparisons as fair as possible, this heuristic algorithm applies elitism by always keeping the best solution found so far.

Figure 5 illustrates the convergence of the GA for the instance of *Type 1* with $\Delta = 0.2$ and $N = 20$. This instance, the most constrained one of *Type 1*, has been selected as representative of the remaining ones, as the GA and the heuristic algorithm present a similar evolution pattern on all of them. The plot corresponds to the evolution of the average population fitness (i.e., the average of the expected total tardiness $E[\widehat{T}]$) across 30 runs both of the GA and the heuristic algorithm. It is clear that the improvement with respect to the best initial solution (the same for both methods) is greater with the GA (45.15% of improvement) than with the heuristic algorithm (12.86%).

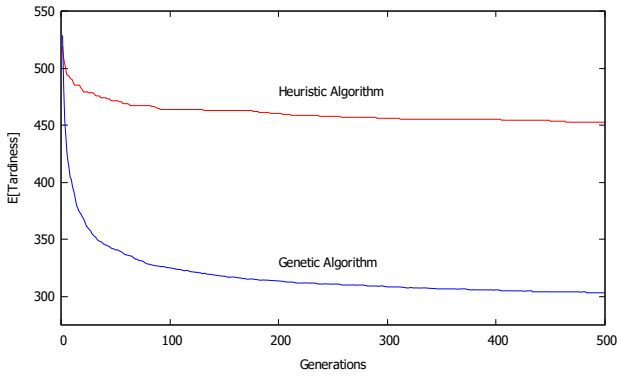


Figure 5: Average fitness evolution for the GA (in blue) and a heuristic algorithm (in red) on the instance of *Type 1* with $N = 20$ and $\Delta = 0.2$.

5.2. Algorithm's performance

To evaluate the GA's performance, since the problem formulation is new to this work and, hence, no other algorithms from the literature exist, we compare the results obtained by the GA with those obtained by the described heuristic algorithm, which essentially consists in applying random or heuristic scheduling rules. The results are shown in Tables 1 to 4, and they correspond to both algorithms running under equivalent conditions (equal number of evaluated individuals). Each row of each table corresponds to one of the benchmark instances, as indicated in the first two columns. For each instance, Tables 1 and 2 give the fuzzy tardiness $\widehat{T}$ together with its expected value $E[\widehat{T}]$ of the best solution found with each algorithm (columns 3,4 for the GA and 5,6 for the heuristic method) for instances of *Type 1* and *Type 2* respectively. All tardiness values are measured in hours. The last column corresponds to the improvement (in percentage) of the GA with respect to the Heuristic. We can see that, on *Type 1* instances, this improvement ranges between 6.2% and slightly more than 69%, with an average improvement near 33%. In fact, an improvement below 10% only takes place for two instances and the average improvement across the remaining instances is in fact greater than 40%. On instances of *Type 2* the improvement of the best solutions is more regular, with an average value of 21.42%. Tables 3 and 4 correspond to the average performance on instances of *Type 1* and *Type 2* respectively. They contain the average expected tardiness of the solution across 30 runs of the GA and the heuristic method together with the standard deviation and the average CPU time (in seconds). The last column reports the average improvement percentage values of the GA with respect to the heuristic method. We can see that, regarding the average expected tardiness, the GA is in average 40.52% better than heuristic algorithm on instances of *Type 1* and 21.38% better on instances of *Type 2* (where the balance between lines becomes more critical), presenting with average values a similar behaviour to the best solution.

To complement the data on the tables, we have run some statistical tests. First, we have run a Wilcoxon signed-rank test for matched samples on the average expected tardiness values obtained by each method across 30 runs on each of the 24 instances. The test confirms that the GA outperforms the heuristic algorithm with a p-value 9.702×10^{-6}.

Secondly, we have performed a more detailed analysis on an instance of each type. We have chosen in each case that instance for which the average percentage improvement of the GA with respect to the heuristic algorithm is closest to the average value across all instances of that type, that is, instance of *Type 1* with $N = 30$ and $\Delta = 0.4$ and instance of *Type 2* with $N = 40$ and $\Delta = 0.4$. For each instance, we have taken the expected tardiness results obtained on the 30 runs of each algorithm and, after checking normality with a Kolmogorov-Smirnoff test, we have run a t-test to compare both algorithms leading to the same conclusion as above with p-values, 3.85×10^{-65} for *Type 1* and 3.35×10^{-69} for *Type 2*. The corresponding boxplots for both instances, in Figure 6, allow to visualise the superiority of the GA.

Table 1: Best total tardiness values for the GA and the heuristic algorithm on instances of *Type 1*.

Instance		GA		Heuristic		
N	Δ	Best $\widehat{T}$	$E[\widehat{T}]$	Best $\widehat{T}$	$E[\widehat{T}]$	% Impr.
20	0.2	(197.7, 278.3, 430.9)	296.3	(335.9, 407.3, 545.7)	424.1	30.12
	0.4	(77.35, 126.3, 244.8)	143.7	(114.1, 165.2, 282.8)	181.8	20.97
	0.6	(62.65, 109.9, 219.8)	125.5	(72.13, 118.6, 227.1)	134.1	6.38
	0.8	(61.41, 109.6, 219.7)	125.1	(72.14, 118.6, 223.9)	133.3	6.19
30	0.2	(28.89, 49.37, 102.7)	57.61	(81.29, 110.3, 177.6)	119.9	51.96
	0.4	(1.300, 8.333, 43.84)	15.45	(8.342, 17.43, 60.75)	25.99	40.54
	0.6	(0.250, 6.616, 42.18)	13.91	(5.483, 12.96, 54.18)	21.39	34.97
	0.8	(0.716, 6.633, 41.64)	13.90	(5.483, 12.96, 54.18)	21.39	35.01
40	0.2	(3.013, 8.470, 27.19)	11.78	(22.83, 34.46, 60.54)	38.07	69.05
	0.4	(0, 0, 0)	0.00	(0, 0, 0)	0.00	-
	0.6	(0, 0, 0)	0.00	(0, 0, 0)	0.00	-
	0.8	(0, 0, 0)	0.00	(0, 0, 0)	0.00	-

Finally, let us notice that the tardiness values on all tables reflect how the difficulty of an instance depends on the (un)even load of EVs on each line L_i and on the values of the parameters N and Δ. As expected, it is easier to find good solutions (with low total tardiness) for instances of *Type 1*, where the load on all lines is even, than on problems of *Type 2*, with uneven loads. It is also clear that problems with a low number of active charging points per line N and strong balance restriction, represented by low values of Δ, are harder to solve. In fact, at the extreme case we find two pathological instances, both with very uneven loads, a small proportion of active charging points per line and under a very strict load balance constraint. These are two instances of *Type 2*, where the first line L_1 supports 60% of the demand, with 108 EVs, 30% (54 EVs) of the demand falls on the second line L_2 and only 10% (18 EVs) falls on the third line L_3. Additionally, these two instances have a small threshold for the maximum number of active charging points $N = 20$ and $N = 30$ (respectively, 33% and 50% of the 60 points available in each line) and a very strict balance constraint with maximum expected imbalance threshold $\Delta = 0.2$. As a result, once the EVs (only 18) that go to the third line L_3 finish charging, the number of EVs that can be expected to be charging at any instant on any of the other two lines is limited to only 4 or 6 (with $N = 20$ and $N = 30$ respectively). This creates a bottleneck and the subsequent delays on the first and second lines, this being specially problematic for L_1, where a total of 90 EVs must be charged. As mentioned above, these instances are pathological and cannot be expected to be solved to satisfaction. They pose instead extreme cases that serve as a challenge for any solving methods that may be proposed for the problem at hand.

Table 2: Best total tardiness values for the GA and the heuristic algorithm on instances of *Type 2*.

Instance		GA		Heuristic		
N	Δ	Best $\widehat{T}$	$E[\widehat{T}]$	Best $\widehat{T}$	$E[\widehat{T}]$	% Impr.
20	0.2	(4481, 4644, 4896)	4666	(5350, 5514, 5757)	5534	15.67
	0.4	(1560, 1707, 1949)	1731	(1961, 2117, 2350)	2137	19.00
	0.6	(701.9, 836.1, 1056)	857.7	(935.2, 1037, 1210)	1054	18.70
	0.8	(372.4, 489.9, 691.1)	510.8	(487.0, 592.4, 778.7)	612.6	16.61
30	0.2	(2473, 2667, 2967)	2694	(3019, 3180, 3418)	3199	15.81
	0.4	(681.2, 804.5, 1000)	822.7	(909.9, 1007, 1174)	1024	19.71
	0.6	(177.5, 252.2, 394.9)	269.2	(270.4, 341.1, 478.9)	357.9	24.78
	0.8	(43.95, 81.29, 182.4)	97.24	(68.12, 114.8, 216.8)	128.6	24.43
40	0.2	(1564, 1705, 1931)	1726	(1961, 2117, 2350)	2137	19.20
	0.4	(302.6, 385.0, 531.6)	401.0	(412.5, 501.3, 652.6)	516.9	22.41
	0.6	(28.84, 64.54, 153.3)	77.81	(58.15, 100.2, 187.7)	111.6	30.29
	0.8	(0.299, 6.307, 57.50)	17.60	(3.506, 15.04, 67.64)	25.30	30.43

6. Conclusions

We have tackled the problem of scheduling the charging of EVs on a real-world charging station. The schedule must take into account some technical constraints of the system, while simultaneously minimise a total tardiness objective function. This work constitutes a first approach to incorporating uncertainty to the problem, to narrow the gap between the model and the real-world situation. In particular, we have considered uncertain charging times modelled as triangular fuzzy numbers. This has resulted in a new formulation of the problem, based on the deterministic one, but with considerable changes, for instance, in the translation of the constraints. Additionally, to solve the resulting problem we have proposed a GA based on a previous algorithm with good performance in the deterministic version. Although the general scheme remains the same, some operators, such as crossover, have had to be adapted to handle fuzzy numbers and a new decoding operator has been proposed, consisting on a purpose-built schedule builder. Finally, we have presented an experimental study on new benchmark fuzzy instances inspired in real-world data that show the correct convergence of the GA as well as its good performance. Additionally, the benchmark instances will be made openly available to the research community on the web, to encourage future advancements on solving this problem.

This constitutes the first approach to the charging problem with uncertainty and as such opens many lines for future research. First, we would like to establish links between the constraint formulations and possibility theory which, in turn, may suggest new and better schedule-building algorithms. It is also possible to give alternative formulations for some constraints in the uncertain setting, in particular, for the balance between lines. Regarding the solving method, it could be improved by combining the GA with local search, which would require defining good neighbourhood structures and efficient neighbour

Table 3: Average expected tardiness of the GA and the heuristic algorithm on instances of *Type 1*.

Instance		GA		Heuristic		
N	Δ	Avg.±SD	CPU(s)	Avg.±SD	CPU(s)	% Impr.
20	0.2	336.0±12.17	40	458.5±7.17	194	29.91
	0.4	149.4±1.39	50	192.3±2.30	199	21.50
	0.6	129.3±0.91	51	137.6±0.66	201	7.00
	0.8	127.1±0.46	55	134.9±0.40	201	6.19
30	0.2	77.74±4.74	54	144.8±5.13	199	52.61
	0.4	17.33±0.46	35	27.29±0.36	206	38.51
	0.6	17.21±0.71	29	23.08±0.36	206	34.04
	0.8	16.66±0.60	30	23.08±0.36	206	34.73
40	0.2	19.41±2.09	44	45.26±2.18	202	64.00
	0.4	0.091±0.03	9	0.137±0.03	206	69.03
	0.6	0.091±0.04	8	0.133±0.03	206	64.33
	0.8	0.091±0.04	8	0.133±0.03	206	64.33

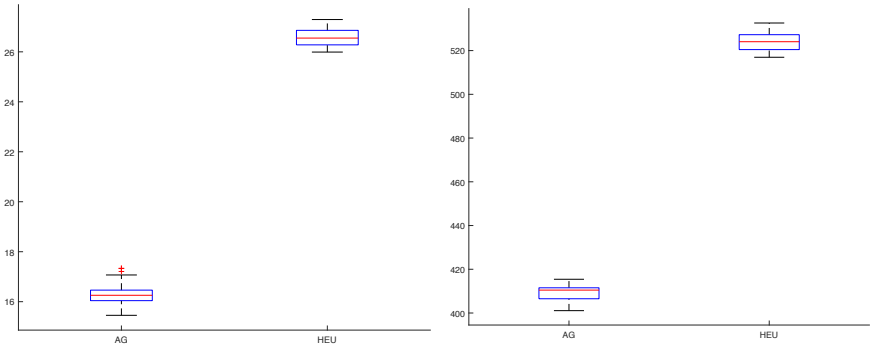
evaluation methods. Another interesting perspective, as considered in Burdett and Kozan (2015), is to consider the robustness of the obtained solutions, either as an objective function to optimise, either as a criterion to compare the fuzzy solutions and the deterministic ones. Finally, we aim at incorporating new features that make the problem closer to the real-life situation, such as having variable energy costs depending on the time of the day.

Acknowledgements

This work was supported by the Spanish Government grant number TIN2016-79190-R.

Table 4: Average expected tardiness of the GA and the heuristic algorithm on instances of *Type 2*.

Instance		GA		Heuristic		
N	Δ	Avg.±SD	CPU(s)	Avg.±SD	CPU(s)	% Impr.
20	0.2	4777±24.00	186	5658±30.41	194	15.60
	0.4	1801±14.58	145	2203±15.55	198	19.10
	0.6	896.7±8.47	111	1085±6.93	200	18.66
	0.8	528.5±4.39	88	631.5±4.78	201	16.56
30	0.2	2772±20.60	173	3343±29.70	196	16.95
	0.4	859.9±9.24	122	1055±6.43	202	19.44
	0.6	279.6±2.53	54	370.8±3.30	207	24.85
	0.8	104.9±1.97	40	136.5±1.87	208	23.50
40	0.2	1794±15.12	148	2186±12.29	198	18.54
	0.4	415.4±3.69	92	532.6±4.17	205	21.87
	0.6	83.86±1.57	42	118.3±1.65	210	30.63
	0.8	20.74±0.69	40	28.19±0.62	210	30.80



Type 1, $N = 30$, $\Delta = 0.4$ *Type 2*, $N = 40$, $\Delta = 0.4$

Figure 6: Boxplot for the results of 30 runs of GA and the heuristic algorithm on two representative instances.

Appendix A.

It is possible to model our problem using an alternative ILP formulation, using the notation summarised in Table A.5, with auxiliary decision variables $\alpha_{ijt}, \beta_{ijt}$ and δ_{ijt} to model the constraint of balance between lines represented in (15), and γ_{ijk}, ρ_{ijk} and ϵ_{ijk} to model the set of overlapping vehicles defined in (11), used in the constraint of maximum active charging points per line. For the experimental results, the time horizon $|H|$ has been calculated as $|H| = \sum_{i,j} p^3_{ij} + \max_{i,j}\{t_{ij}\}$, which provides an upper bound on the total completion time.

Table A.5: Notation used in the ILP formulation for problem parameters

Parameters	
$i \in \{1,2,3\}$	charging lines
$j \in M_i$	number of vehicles arriving at each line
$l \in \{1,2,3\}$	components of a TFN
N	maximum number of charging points that can be active simultaneously
v_{ij}	vehicle that arrives at j^{th} order to line i
t_{ij}	arrival time of vehicle v_{ij}
$\widehat{p_{ij}}$	fuzzy processing time of vehicle v_{ij}
d_{ij}	departure time of the vehicle given by driver
Δ	maximum load imbalance between any two lines
M	a sufficiently large number for disjunctive constraints taken as $2 \times \|H\|$ in experiments
$t \in H$	discrete time periods within a horizon H
Decision variables	
$\widehat{T_{ij}} = (T^1_{ij}, T^2_{ij}, T^3_{ij})$	fuzzy tardiness of vehicle v_{ij}, $T^l_{ij} \in \mathbb{Z}$
$\widehat{st_{ij}} = (st^1_{ij}, st^2_{ij}, st^3_{ij})$	fuzzy charging starting time for vehicle v_{ij}, $st^l_{ij} \in \mathbb{Z}$
α_{ijt}	binary variable, equal to 1 if v_{ij} is expected to have started charging before time t
β_{ijt}	binary variable, equal to 1 if v_{ij} is expected to continue charging after time t
δ_{ijt}	binary variable, equal to 1 if v_{ij} is expected to be charging at time t
γ_{ijk}	binary variable, equal to 1 if v_{ik} might have started charging before v_{ij} starts charging
ρ_{ijk}	binary variable, equal to 1 if v_{ik} might finish charging after v_{ij} starts charging
ϵ_{ijk}	binary variable, equal to 1 if v_{ik} might be charging when v_{ij} starts charging

$$\min_{\widehat{st},\widehat{T},\alpha,\beta,\delta,\gamma,\rho,\epsilon} \sum_{i,j} \frac{T^1_{ij} + 2T^2_{ij} + T^3_{ij}}{4}$$

s.t.

$$T^l_{ij} \geq st^l_{ij} + p^l_{ij} - d_{i,j} \qquad \forall i,j,l \tag{A.1}$$

$$t - \frac{(st^1_{ij} + 2st^2_{ij} + st^3_{ij})}{4} + 1 \leq M\alpha_{ijt} \qquad \forall i,j,t \tag{A.2}$$

$$\frac{st^1_{ij} + 2st^2_{ij} + st^3_{ij}}{4} - t \leq M(1 - \alpha_{ijt}) \qquad \forall i,j,t \tag{A.3}$$

$$t - \frac{(st^1_{ij} + 2st^2_{ij} + st^3_{ij} + p^1_{ij} + 2p^2_{ij} + p^3_{ij})}{4} \leq M(1 - \beta_{ijt}) \qquad \forall i,j,t \tag{A.4}$$

$$\frac{st^1_{ij} + 2st^2_{ij} + st^3_{ij} + p^1_{ij} + 2p^2_{ij} + p^3_{ij}}{4} - t + 1 \leq M\beta_{ijt} \qquad \forall i,j,t \tag{A.5}$$

$$\delta_{ijt} = \alpha_{ijt} + \beta_{ijt} - 1 \qquad \forall i,j,t \tag{A.6}$$

$$\sum_{j \in M_i} \delta_{ijt} - \sum_{j \in M_{i'}} \delta_{i'jt} \leq \Delta N \qquad \forall i,i' \in \{1,2,3\}, \forall t \tag{A.7}$$

$$st^1_{ij} \geq t_{ij} \qquad \forall i,j \tag{A.8}$$

$$st^l_{ij} - st^l_{ik} + 1 \leq M\gamma_{ijk} \qquad \forall i, \forall j,k \in M_i, \forall l \tag{A.9}$$

$$st^l_{ik} + p^l_{ik} - st^l_{ij} \leq M\rho_{ijk} \qquad \forall i, \forall j,k \in M_i, \forall l \tag{A.10}$$

$$\epsilon_{ijk} = \gamma_{ijk} + \rho_{ijk} - 1 \qquad \forall i, \forall j,k \in M_i \tag{A.11}$$

$$\sum_{k \in M_i} \epsilon_{ijk} \leq N \qquad \forall i,j \tag{A.12}$$

$$0 \leq st^1_{ij} \leq st^2_{ij} \leq st^3_{ij} \qquad \forall i,j \tag{A.13}$$

$$0 \leq T^1_{ij} \leq T^2_{ij} \leq T^3_{ij} \qquad \forall i,j \tag{A.14}$$

$$\alpha_{ijt}, \beta_{ijt}, \delta_{ijt} \in \{0,1\} \qquad \forall i,j,t \tag{A.15}$$

$$\gamma_{ijk}, \rho_{ijk}, \epsilon_{ijk} \in \{0,1\} \qquad \forall i, \forall j,k \in M_i \tag{A.16}$$

$$st^l_{ij}, T^l_{ij} \in \mathbb{Z} \qquad \forall i,j,l \tag{A.17}$$

Regarding the constraints in the ILP model, (A.1) corresponds to minimising the tardiness of each vehicle, as implicitly defined in (16). (A.2) and (A.3) ensure that α_{ijt} takes value 1 if time $t \geq E[\widehat{st_{ij}}]$ and 0 otherwise; similarly, (A.4) and (A.5) ensure that β_{ijt} takes value 1 if time $t \leq E[\widehat{st_{ij}} + \widehat{p_{ij}}]$ (i.e. $t \leq E[\widehat{c_{ij}}]$) and 0 otherwise and (A.6) causes δ_{ijt} to be 1 if α_{ijt} and β_{ijt} are both 1, and 0 otherwise (notice that it cannot be -1 since α and β cannot be both 0 at the same time). In consequence, (A.7) ensures that the load imbalance between any two lines does not exceed Δ (constraint (13) in the original model). Constraint (A.8) is the same as (9) for the EV's arrival time. (A.9) ensures that γ_{ijk} takes value

1 if vehicle v_{ik} might have started charging before vehicle v_{ij} and 0 otherwise, while (A.10) ensures that ρ_{ijk} takes value 1 if vehicle v_{ik} might be still charging when vehicle v_{ij} starts charging and 0 otherwise. Thus, (A.11) causes ϵ_{ijk} to take value 1 if vehicles v_{ik} and v_{ij} might be charging simultaneously, so (A.12) ensures that there cannot be more than N vehicles charging at the same time on the same line when vehicle v_{ij} might start charging. This corresponds to the constraint of maximum active charging points per line modelled by (11) and (12). Finally, (A.13) and (A.14) ensure the right order of the three TFN components and nonnegativity, (A.15) and (A.16) model integrality constraints of binary variables and (A.17) states that st_{ij}^{l} and T_{ij}^{l} are all integer variables.

References

Abdullah, S., Abdolrazzagh-Nezhad, M., 2014. Fuzzy job-shop scheduling problems: A review. Information Sciences 278, 380–407. doi:10.1016/j.ins.2014.03.060.

Ansari, M., Al-Awami, A.T., Sortomme, E., Abido, M.A., 2015. Coordinated bidding of ancillary services for vehicle-to-grid using fuzzy optimization. IEEE Transactions on Smart Grid 6, 261–270. doi:10.1109/TSG.2014.2341625.

Balin, S., 2011. Parallel machine scheduling with fuzzy processing times using a robust genetic algorithm and simulation. Information Sciences 181, 3551–3569.

Baudrit, C., Dubois, D., 2006. Practical representations of incomplete probabilistic knowledge. Computational Statistics & Data Analysis 51, 86–108.

Brunelli, M., Mezei, J., 2013. How different are ranking methods for fuzzy numbers? A numerical study. International Journal of Approximate Reasoning 54, 627–639. doi 10.1016/j.ijar.2013.01.009

Burdett, R., Kozan, E., 2015. Techniques to effectively buffer schedules in the face of uncertainties. Computers & Industrial Engineering 87, 16–29.

Chanas, S., Kasperski, A., 2001. Minimizing maximum lateness in a single machine scheduling problem with fuzzy processing times and fuzzy due dates. Engineering Applications of Artificial Intelligence 14, 377–386.

Chanas, S., Nowakowski, M., 1988. Single value simulation of fuzzy variable. Fuzzy Sets and Systems 25, 43–57.

Destercke, S., Couso, I., 2015. Ranking of fuzzy intervals seen through the imprecise probabilistic lens. Fuzzy Sets and Systems 278, 20–39. doi 10.1016/j.fss.2014.12.009.

Dubois, D., 2006. Possibility theory an statistical reasoning. Computational Statistics & Data Analysis 51, 47–69.

Dubois, D., 2011. The role of fuzzy sets in decision sciences: Old techniques and new directions. Fuzzy Sets and Systems 184, 3–28. doi:10.1016/j.fss.2011.06.003.

Dubois, D., Fargier, H., Fortemps, P., 2003. Fuzzy scheduling: Modelling flexible constraints vs. coping with incomplete knowledge. European Journal of Operational Research 147, 231–252.

Dubois, D., Fargier, H., Fortemps, P., 2008. Scheduling under flexible constraints and uncertain data: the fuzzy approach, in: Production Scheduling. Wiley. chapter 11, pp. 301–332.

Dubois, D., Foulloy, L., Mauris, G., Prade, H., 2004. Probability-possibility transformations, triangular fuzzy sets and probabilistic inequalities. Reliable Computing 10, 273–297.

Dubois, D., Prade, H., 1986. Possibility Theory: An Approach to Computerized Processing of Uncertainty. Plenum Press, New York (USA).

Dubois, D., Prade, H., 1987. The mean value of a fuzzy number. Fuzzy Sets and Systems 24, 279–300.

Fortemps, P., 1997. Jobshop scheduling with imprecise durations: a fuzzy approach. IEEE Transactions of Fuzzy Systems 7, 557–569.

García-Álvarez, J., González, M., Vela, C., 2015. A genetic algorithm for scheduling electric vehicle charging, in: Proceedings of the Genetic and Evolutionay Computation Conference (GECCO 2015), pp. 393–400.

Ghrayeb, O.A., 2003. A bi-criteria optimization: minimizing the integral value and spread of the fuzzy makespan of job shop scheduling problems. Applied Soft Computing 2, 197–210.

Guo, S., Zhao, H., 2015. Optimal site selection of electric vehicle charging station by using fuzzy TOPSIS based on sustainability perspective. Applied Energy 158. doi:10.1016/j.apenergy.2015.08.082.

Hajforoosh, S., Masoum, M.A., Islam, S.M., 2015. Real-time charging coordination of plug-in electric vehicles based on hybrid fuzzy discrete particle swarm optimization. Electric Power Systems Research 128, 19–29. doi:10.1016/j.epsr.2015.06.019.

Hapke, M., Slowinski, R., 1996. Fuzzy priority heuristics for project scheduling. Fuzzy Sets and Systems 83, 291–299. doi:10.1016/0165-0114(95)00338-X.

Heilpern, S., 1992. The expected value of a fuzzy number. Fuzzy Sets and Systems 47, 81–86.

Hernandez-Arauzo, A., Puente, J., Varela, R., Sedano, J., 2015. Electric vehicle charging under power and balance constraints as dynamic scheduling. Computers & Industrial Engineering 85, 306–315. doi:10.1016/j.cie.2015.04.002.

IBM, 2014. IBM CPLEX Optimizer. URL: http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/index.html.

Kang, J., Duncan, S.J., Mavris, D.N., 2013. Real-time scheduling techniques for electric vehicle charging in support of frequency regulation. Procedia Computer Science 16, 767–775. doi:10.1016/j.procs.2013.01.080.

Kuroda, M., Wang, Z., 1996. Fuzzy job shop scheduling. International Journal of Production Economics 44, 45–51.

Lei, D., 2010. Solving fuzzy job shop scheduling problems using random key genetic algorithm. International Journal of Advanced Manufacturing Technologies 49, 253–262.

Liu, B., Fan, Y., Liu, Y., 2015. A fast estimation of distribution algorithm for dynamic fuzzy flexible job-shop scheduling problem. Computers & Industrial Engineering 87, 193–201.

Mauris, G., Lasserre, V., Foulloy, L., 2001. A fuzzy approach for the expression of uncertainty in measurement. Measurement 29, 165–177. doi:10.1016/S0263-2241(00)00036-1.

Palacios, J.J., González, M.A., Vela, C.R., González-Rodríguez, I., Puente, J., 2015a. Genetic tabu search for the fuzzy flexible job shop problem. Computers & Operations Research 54, 74–89. doi:10.1016/j.cor.2014.08.023.

Palacios, J.J., González-Rodríguez, I., Vela, C.R., Puente, J., 2015b. Coevolutionary makespan optimisation through different ranking methods for the fuzzy flexible job shop. Fuzzy Sets and Systems 278, 81–97. doi:10.1016/j.fss.2014.12.003.

Palacios, J.J., González-Rodríguez, I., Vela, C.R., Puente, J., 2017. Robust multiobjective optimisation for fuzzy job shop problems. Applied Soft Computing 56, 604–616. doi 10.1016/j.asoc.2016.07.004

Rahman, I., Vasant, P., Singh, B., Abdullah-Al-Wadud, M., Adnan, N., 2016. Review of recent trends in optimization techniques for plug-in hybrid, and electric vehicle charging infrastructures. Renewable and Sustainable Energy Reviews 58, 1039–1047. doi:10.1016/j.rser.2015.12.353

Rommelfanger, H., 1990. FULPAL — an interactive method for solving (multiobjective) fuzzy linear programming problems, in: Slowinski, R., Teghem, J. (Eds.), Stochastic Versus Fuzzy Approaches to Multiobjective Mathematical Programming under Uncertainty. Springer, pp. 279–299. doi:10.1007/978-94-009-2111-5_14.

Sedano, J., Portal, M., Hernández-Arauzo, A., Villar, J.R., Puente, J., Varela, R., 2013. Intelligent system for electric vehicle charging: Design and operation. DYNA 88, 644–651. doi:10.6036/5788.

Talbi, E.G., 2009. Metaheuristics. From Design to Implementation. Wiley.

Wang, B., Li, Q., Yang, X., Wang, X., 2010. Robust and satisfactory job shop scheduling under fuzzy processing times and flexible due dates, in: Proc. of the 2010 IEEE International Conference on Automation and Logistics, pp. 575–580.

Wong, B.K., Lai, V.S., 2011. A survey of the application of fuzzy set theory in production and operations management: 1998–2009. International Journal of Production Economics 129, 157–168.

You, P.S., Hsieh, Y.C., 2014. A hybrid heuristic approach to the problem of the location of vehicle charging stations. Computers & Industrial Engineering 70, 195–204.

Zhong, S., Chen, Y., Zhou, J., 2015. Fuzzy random programming models for location-allocation problem with applications. Computers & Industrial Engineering 89, 194–202.

8.3. Electric vehicle charging scheduling by an enhanced artificial bee colony algorithm

Presentamos la siguiente publicación:

- **Título:** Electric vehicle charging scheduling by an enhanced artificial bee colony algorithm.
- **Revista:** *Energies*
- **Año:** 2018.
- **DOI:** 10.3390/en11102752.
- **Referencia:** García Álvarez, J.; González, M.A.; Rodríguez Vela, C.; Varela, R. Electric Vehicle Charging Scheduling by an Enhanced Artificial Bee Colony Algorithm. *Energies*. 2018, 11, 2752. DOI: 10.3390/en11102752.

energies

MDPI

Article

Electric Vehicle Charging Scheduling by an Enhanced Artificial Bee Colony Algorithm †

Jorge García Álvarez, Miguel Ángel González, Camino Rodríguez Vela and Ramiro Varela *

Department of Computer Science, University of Oviedo, 33204 Gijón, Spain;
jgarcia-alvarez@outlook.com (J.G.Á.); mig@uniovi.es (M.Á.G.); crvela@uniovi.es (C.R.V.)

* Correspondence: ramiro@uniovi.es; Tel.: +34-985-182-508

† This paper is an extended version of our paper published in: García-Álvarez, J.; González, M.; Vela, C.; Varela, R. Electric Vehicle Charging Scheduling Using an Artificial Bee Colony Algorithm. In *Ferrández Vicente J., Álvarez-Sánchez J., de la Paz López F., Toledo Moreo J., Adeli H. (eds) Natural and Artificial Computation for Biomedicine and Neuroscience. IWINAC 2017*. Lecture Notes in Computer Science, vol. 10337. pp. 115–124. Springer, Cham.

Received: 11 September 2018; Accepted: 11 October 2018; Published: 14 October 2018

Abstract: Scheduling the charging times of a large fleet of Electric Vehicles (EVs) may be a hard problem due to the physical structure and conditions of the charging station. In this paper, we tackle an EV's charging scheduling problem derived from a charging station designed to be installed in community parking where each EV has its own parking lot. The main goals are to satisfy the user demands and at the same time to make the best use of the available power. To solve the problem, we propose an artificial bee colony (ABC) algorithm enhanced with local search and some mating strategies borrowed from genetic algorithms. The proposal is analyzed experimentally by simulation and compared with other methods previously proposed for the same problem. The results of the experimental study provided interesting insights about the problem and showed that the proposed algorithm is quite competitive with previous methods.

Keywords: electric vehicle charging; scheduling; artificial bee colony; local search; metaheuristics

1. Introduction

Electric Vehicles (EVs) are increasingly important nowadays for environmental reasons. However, their sustainable deployment requires new technology and infrastructures as, for example, specialized charging stations to accommodate large fleets of EVs. There have been considerable research effort on different aspects related to EV charging as, for example, the location of EV charging stations [1–4] or electricity price forecasting [5], among others. Charging scheduling is particularly interesting due to the large charging times often required by batteries and also to the power and physical constraints of the charging stations. Only with the use of smart scheduling algorithms is it possible to make the best use of the available resources and to satisfy the user requirements at the same time [6,7].

In this paper, we tackle the problem raised in [8], which is motivated by a charging station designed to be exploited in a residential parking where each vehicle must be parked in its owner's space. This physical constraint, together with others derived from the electric characteristics of the charging station, makes the problem of scheduling the EV's charging times over periods of large demand difficult. For this reason, a number of metaheuristics were proposed to solve the problem, in particular dispatching rules [8], memetic algorithm [9] and artificial bee colony (ABC) algorithm [10].

In this paper, we build on the work presented in [10] and include a number of new contributions, in addition to a thorough updating of the literature review: (1) We introduce several new strategies for both employed and onlooker bee phases, which contribute to improving the performance of the

ABC algorithm so that it is able to reach much better solutions; (2) We hybridize the algorithm with an additional local search step, which allows the algorithm to further improve the final solution; (3) We performed a much more comprehensive experimental study using a larger set of instances than that used in previous studies, in which we compare our proposals and show that the new hybrid algorithm was able to improve the quality of the results reported in [8–10].

The remainder of this paper is organized as follows: in Section 2, we review the literature on EV charging scheduling. In Section 3, we present the characteristics of the charging station considered herein. Section 4 defines the scheduling problem. In Section 5, we describe the artificial bee colony algorithm and the proposed enhancements. The results of the experimental study are reported and analyzed in Section 6. Finally, Section 7 summarizes the main contributions of this paper and proposes some ideas for further work.

2. Literature Review

EV charging raises challenging scheduling problems. Indeed, the research community has proposed a number of EV charging scheduling models in the last years. Some comprehensive reviews of such problems and solving techniques are given in [6,7,11].

There are two main strategies for scheduling EV charging. The first one is distributed or decentralized strategy [12–14], which offers great flexibility to EV owners, allowing them to decide about when and how to charge their EVs. This may give rise to some security problems and overload in the power grid. Some of these inconveniences may be avoided to some extent by imposing some price incentives that stimulate EV owners to charge their EVs in off-peak hours (see, for example, [15,16]). However, it is generally difficult to achieve an optimal charging strategy [17].

The second strategy is centralized charging, in which an aggregator makes all the scheduling decisions as starting time and charging rate of the EVs. In order to build a schedule, the aggregator receives the status of all EVs along with their desired departure times (provided by EV owners) and tries to optimize both the customers satisfaction and the use of the available power [18]. Most papers in the literature recommend this strategy over the decentralized one. Indeed, some studies demonstrate that a coordinated charging scheme with some valley-filling strategy significantly outperforms uncoordinated charging by suppressing elevated peak load demands on the power grid [19].

Regarding the objectives pursued, there are a number of strategies. Some papers perform single objective optimization of total cost [20,21], total tardiness [22] or the use of renewable energy sources [23]. It is also usual to minimize peak demands or grid congestion [17,24]. Other approaches consider three-phase power flows and try to minimize the imbalance between phases [8,25]. Clearly, in these types of problems, there may be many relevant objectives, and therefore multi-objective optimization is often considered. For example, in [26], the optimization of the total load variance and EV owners' preferences is considered. Another interesting example can be found in [27], where the objectives are the minimization of the costs incurred from being parked, maximization of the revenues offering secondary regulation and the maximization of the vehicle fleet charging station efficiency.

It is important to mention that each environment and EV charging station present particular characteristics, so each of them gives rise to a different model and poses its own scheduling problem. For example, some systems consider varying the charging rate of the EVs [24], varying power of the charging station [28], or varying electricity prices [29,30]. Other models consider uncertainty and stochastic parameters [31,32]. Another issue is whether [33] or not [8] the scheduler can decide to which dock each EV is assigned. Some models even consider charging/discharging strategies to optimize the overall process [34].

Due to the complexity of these problems, many existing optimization techniques have been applied to solve them—for example, linear programming [35], ant-based swarm algorithm [12], genetic algorithm [27], particle swarm optimization [21,36], multi-agent based system [37], estimation of distribution algorithm [30], moving window optimization [29], tabu search and GRASP [38], among others.

As mentioned, the model considered in this work is that proposed in [8]. This model corresponds to the charging station described in [39], which is designed to be installed in community parking. In this station, a parking lot has a charging point connected to one of the lines of a three-phase feeder. This, together with the fact that each user must use his own parking lot, gives rise to the main constraint of the model: the load imbalance on the three lines must be limited. The solution proposed in [8] relies on the use of dispatching rules to meet online scheduling requirements. For this reason, the quality of the solutions is moderate and may be improved as demonstrated in further studies [22] by means of offline algorithms. Hence, two new approaches were proposed in order to obtain better solutions, memetic algorithm [9] and artificial bee colony algorithm [10]. We argue that the results of the later could still be improved with some intensification step. To this end, we propose incorporating problem domain knowledge by local search and some other improvements in different phases of the artificial bee colony algorithm.

3. The Charging Station

The design and operation modes of the charging station is detailed in [39]; Figure 1 shows its general structure. The station is fed by three-phase electric power with voltage between phases of 400 V. Each charging point is connected to one single-phase at 230 V and 7.3 kW (32 A). In principle, we follow a so-called Mode 3 in the regulation UNE-EN 61851-1 [40], which considers charging at a constant rate. As pointed in [41], this is the most suitable method for domestic environments.

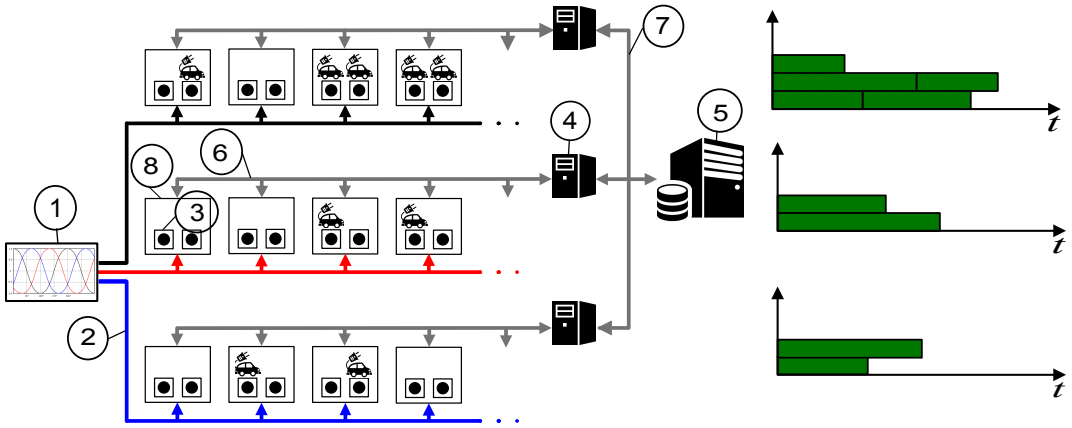


Figure 1. Components of the charging station: (1) power source; (2) three-phase electric power; (3) charging points; (4) masters; (5) server with database; (6) asynchronous serial connections; (7) communication TCP/IP (Transmission Control Protocol/Internet Protocol); (8) slaves. The Gantt Chart on the right shows the charging intervals of the vehicles in each line in a feasible schedule for $n = 3$ and $\Delta = 1/3$.

As shown in Figure 1, the station is controlled by a server together with a number of masters and slaves. Each slave controls two charging points of type 2/AC IEC 62196-2. A master is connected to eight slaves and has a user interface. The server centralizes the control and receives signals from the slaves regarding events as connection or disconnection of EVs. The server also sends orders to the slaves to activate and deactivate charging points in accordance with the schedule.

Even though there are many spaces available (180 in our experimental study), not all the charging points in these spaces can be active at the same time due to the available power being limited. For example, if electricity supply of 50 kW three phase power (3/AC) were contracted and each EV requires 7.3 kW, at most 21 EVs can be charging simultaneously in a line at the maximum power. To avoid energy losses, the imbalance among the three lines must be limited. To this end, a hard maximum imbalance between every two lines is established.

4. Problem Definition

As it was done in [10], we consider the static and dynamic versions of the problem. The first one is not realistic, but it helps to better understand the problem and to establish performance limits to the solution to the dynamic problem, which models the real scenario. Finally, we comment on some model extensions.

4.1. The Static Problem

We have three lines $L_i, 1 \leq i \leq 3$ with n_i charging points each. Each line L_i receives M_i vehicles, denoted by $v_{i1}, \ldots, v_{iM_i}$. We denote by n the total number of vehicles, i.e., $n = M_1 + M_2 + M_3$. For each vehicle, we are given the arrival time $t_{ij} \geq 0$, the charging time $p_{ij} > 0$ and the due date $d_{ij} \geq t_{ij} + p_{ij}$, namely the time at which the user is expected to take the vehicle away. In this static version, it is assumed that all these data are known in advance at the beginning of the scheduling horizon ($t = 0$).

The objective is to build up a feasible schedule; i.e., to establish starting times st_{ij}, $1 \leq i \leq 3$, $1 \leq j \leq M_i$, for charging the EVs so that the following constraints are satisfied:

$$\forall v_{ij}, \quad st_{ij} \geq t_{ij}, \tag{1}$$

$$\forall v_{ij}, \quad C_{ij} = st_{ij} + p_{ij}, \tag{2}$$

$$N_i(t) \leq N, \quad t \geq 0; 1 \leq i \leq 3, \tag{3}$$

$$\frac{\left|N_i(t) - N_j(t)\right|}{N} \leq \Delta, \quad t \geq 0; 1 \leq i, j \leq 3, \tag{4}$$

where C_{ij} is the completion time of charge of the EV v_{ij}, n is a parameter that fixes the maximum number of EVs that can be charging in a line at the same time (due to the contracted electricity supply, as described in Section 3), $N_i(t)$ is the number of EVs charging at the same time in the line i over the interval $[t, t+1)$ and $\Delta \in [0, 1]$ is a parameter.

Equation (1) forbids EVs to start charging before their arrival time. Equation (2) ensures that EVs cannot be disconnected until they finish charging. Equation (3) establishes that the number of EVs charging in a line cannot exceed n . Finally, Equation (4) establishes the maximum imbalance between any two lines by means of the parameter Δ.

The objective function is the total tardiness defined as:

$$\sum_{i=1}^{3} \sum_{j=1}^{M_i} \max(0, C_{ij} - d_{ij}), \tag{5}$$

which should be minimized.

The right-hand side of Figure 1 shows the Gantt chart of a possible schedule for the small problem represented in the charging station on the left-hand side, larger examples of feasible schedules for real-size instances with 180 vehicles are shown in Section 6.

4.2. The Dynamic Problem

In the dynamic problem, we do not know in advance the due dates, charging and arrival times of the vehicles. Therefore, following [8], it is modeled as a sequence $P_1, P_2, \ldots$ of static instances at times $T_1, T_2, \ldots$. The instance P_k is given by the EVs in the system arriving by T_k that have not completed charging. Some of them have started to charge before T_k and we know their completion times C_{ij}, while others have not yet started to charge. For these last ones, we know the charging times p_{ij} and due dates d_{ij}. The goal is to obtain a feasible schedule for these EVs that minimize the total tardiness and that satisfy all the constraints naturally derived from the static problem.

At each time point T_k, a supervisor program running on the server checks for new EV arrivals since $T_{k-1} = T_k - \Delta T$. If some EV arrived, a new instance P_k is created and solved; otherwise, the current

schedule remains valid until the next time point. Notice that the st_{ij} may be modified in further P_k instances as long as v_{ij} does not start charging. The time interval ΔT is set at two minutes in order to not overload the server in situations where many EVs arrive at almost the same time and also to make the best use of the charging resources. For more detailed definition of the dynamic problem, we refer the interested reader to [8].

4.3. Model Extensions

In previous descriptions of the charging model, we have done some simplifying assumptions for the sake of clarity, namely, all EVs charge at the same constant rate, the user never takes the EV away before the completion time of charging, the batteries never get fully charged before this time, and the contracted power in the charging station is constant over time. However, the model may be extended to deal with these situations as well. For example, variable contracted power may be modeled by considering that the capacity of each line is organized into slots whose number varies over time. Therefore, charging at a variable rate is possible by assigning an EV a different number of slots at different time intervals. In addition, situations such as EVs going out or EVs completing charging before their charging times may be managed by events that the slaves can identify and communicate to the server, much in the same way as when a new EV arrives. With all of these, the energy requirement of an EV should be given as the number of time slots required to reach the desired state of charge of the battery. Of course, with these extensions, the scheduling problem will be harder to solve, but the proposed algorithms could be naturally adapted.

5. Artificial Bee Colony Algorithm

The Artificial Bee Colony algorithm (ABC) is a swarm population-based metaheuristic introduced in [42], which is inspired by the intelligent foraging behavior of honey bees. The method mimics the search for food of three types of foraging bees: employed, onlooker and scout. ABC is often used to solve scheduling problems because of its effectiveness and its good balance between diversification and intensification. A review of its fundamentals and some applications can be found in [43]. The ABC algorithm was proposed for numerical optimization and so, in principle, the food sources should be encoded by vectors of real numbers in a given interval. However, unlike other similar metaheuristics as Particle Swarm Optimization (PSO), whose variation operators strongly rely on accurate distance metrics between solutions, ABC is easy to adapt to discrete; i.e., combinatorial, optimization. This was done, for example, in [44] where the authors tackled a variant of the Flow Shop Scheduling problem and encode food sources by permutations of jobs. We use similar coding schema for the EV charging scheduling problem herein.

As shown in Algorithm 1, the proposed ABC algorithm starts creating a number of SN initial solutions or food sources. Then, it iterates over a number of cycles. In each cycle, several steps are performed: employed bee phase, onlooker bee phase and scout bee phase. The termination criterion is satisfied when the best solution is not improved for a consecutive number of cycles, or if a solution with zero tardiness is reached. Finally, local search is applied to the best solution found so far. The main features and steps of this algorithm are described below.

Algorithm 1 The Artificial Bee Colony algorithm

Input An EV charging scheduling problem instance P
  Generate the initial population;
  while Termination criterion is not satisfied **do**
    Apply employed bee phase;
    Apply onlooker bee phase;
    Apply scout bee phase;
  end while
  Apply local search to the best solution;
  return The schedule from the best solution reached;

5.1. Food Source Representation and Evaluation

We encode solutions; i.e., food sources, as permutations of EVs. In the static problem, all EVs of the problem are considered as the algorithm is applied only once to solve the problem. However, in the dynamic problem, the algorithm is applied to solve each instance P_k and so the permutation contains only the EVs at T_k that do not start to charge. Each solution has an associated value, num_trials, which is the number of times it was tried to improve without success.

To evaluate a food source, a schedule S is built from the permutation by sequentially scheduling the EVs at the earliest possible starting time such that all the constraints defined in Section 4 are met with respect to the EVs previously scheduled. The amount of nectar (the measure of quality of a solution) is the value $1/tard_S$, where $tard_S$ is the tardiness of the solution calculated by Equation (5). Abusing the language, in the following, we will use the symbol S to denote both the schedule and the permutation of EVs corresponding to a given solution.

5.2. Initial Population

We propose combining two dispatching rules with some random food sources to create an initial population. The goal is to achieve good balance among quality and diversity. The first dispatching rule is the Due Date Rule (DDR), which sorts all vehicles in increasing order of their due dates d_{ij}. The second is the Latest Starting Time (LST) rule, which sorts all vehicles in increasing order of their latest starting times, defined as $lst_{ij} = d_{ij} - p_{ij}$. Both rules are deterministic, so we follow the approach proposed in [22] to create diverse food sources. To add the next vehicle to the permutation V, we sort the vehicles not yet added to V using the corresponding dispatching rule and then we perform a tournament selection: a number t_{size} of vehicles is selected uniformly and the best of them according to the ordering given by the dispatching rule is added to V. As we will see, the parameter t_{size} is actually relevant: too large values generate too similar food sources, whereas too small values generate almost random ones. To build up the initial population, one third of the individuals are generated by each dispatching rule and the other third at random.

5.3. Employed Bee Phase

Employed bees are in charge of searching for new and hopefully better food sources. To this end, in the original ABC algorithm, each employed bee generates one new candidate solution in the neighborhood of the solution in one food source. The new candidate replaces the old solution if it is better. In our algorithm, we propose exploiting crossover operators borrowed from genetic algorithms following two different approaches.

In the first one (denoted e_meth1), the best solution found so far is selected, unless it was already selected in previous cycles. In that case, we choose the food source with the largest num_trials such that it was never chosen for this role. This requires maintaining the list "common parents" containing the solutions that were already chosen. Then, the food source in each employed bee is combined with the selected food source, generating two new offspring. The best of them replaces the food source in the memory of the employed bee if it is better; in that case, num_trials is set to zero for this food source; otherwise, the original food source remains in the population with num_trials increased in one unit. The rationale of this method is that it may produce good solutions as a solution that was not improved after many trials may be a good solution. At the same time, it may produce reasonable diversity, as new solutions are always created from different outstanding food sources in successive cycles of the algorithm.

In the second approach (denoted e_meth2), we randomly shuffle the food sources of the population and organize them in pairs to be combined. Thus, all food sources in the population are combined. In this way, the diversity is expected to be larger than in the first method at the cost of lower quality.

We consider different crossover operators to combine solutions. The first one, which is specially designed for this problem, is the Starting-time Based Crossover (SBX), initially proposed in [22].

It randomly selects a time $t_0 \in [T_{min}, T_{max}]$ (where T_{min} is the minimum starting time of all vehicles in both parents and T_{max} is the maximum starting time), and builds the first offspring O_1 with all the EVs in the first parent P_1 that are scheduled before t_0, in the order they appear in P_1, followed by the remaining EVs in the same order as they appear in P_2. The second offspring O_2 is created similarly by exchanging the roles of P_1 and P_2. The second operator is the classic Partially-Mapped Crossover (PMX) proposed in [45], which is commonly used in permutations encoding. In addition, we consider a third option that consists of choosing SBX or PMX at random each time. In [10], it was proven that this method obtains better results than using each operator separately. In Section 6, we report results from some experiments carried out to compare all these options.

5.4. Onlooker Bee Phase

Employed bees share their information with onlooker bees waiting in the hive. Then, onlooker bees probabilistically choose their food sources to go. Once there, they try to find a better neighboring source. In particular, the probability to choose a food source k in this phase is

$$prob_k = (1/tard_k) / \sum_{j=1}^{SN} 1/tard_j. \quad (6)$$

Notice that, in this case, divisions by zero will not be produced as the algorithm ends as soon as a solution with zero tardiness is reached.

We propose different ways to apply the onlooker bee phase, which are adapted to the EV charging scheduling problem. The rationale is trying to schedule a vehicle with tardiness earlier or to delay a vehicle without tardiness.

The first one, termed *o_meth*1, is a generalization of the procedure proposed in [10], which selects at random up to the 10% of the EVs in the permutation $(v_1, \ldots, v_n)$. For each selected EV, v_i, its tardiness is checked. If it is zero, v_i could possibly be delayed; therefore we try to swap v_i with all the EVs from its position onwards, until an improving solution is found. However, if the tardiness is positive v_i is swapped with the previous ones instead. In any case, as soon as an improving solution is reached, it replaces the original one and *num_trials* is set to zero. Otherwise, the original solution remains and *num_trials* is increased in one unit.

In this paper, we enhance this procedure by using two parameters: *max_improv* and *step_size*. As soon as we find a swap that leads to a better solution, we set $num\_improv \leftarrow num\_improv + 1$, and we repeat the process from this new solution, unless we have reached the maximum of *max_improv* improvements. On the other hand, *step_size* is used so that not all swaps are tried, but only a swap for every *step_size* vehicles. This new proposal may be better than that in [10] because the parameter *max_improv* allows for increasing the intensification (of course at the cost of increasing the computational time), whereas the parameter *step_size* allows the algorithm to reduce the computational time, as it avoids testing all candidate swaps. The enhanced procedure is denoted *o_meth*1 and it is showed in Algorithm 2. The proposal described in [10] would be a particular case taking $max\_improv = 1$ and $step\_size = 1$.

Algorithm 2 First type improvement for the onlooker bee phase *o_meth*1

Input A solution S and parameters *step_size* and *max_improv*
  chosen_vehicles $\leftarrow 0$;
  num_improv $\leftarrow 0$;
  while *chosen_vehicles* $< n * 0.1$ **and** *num_improv* $<$ *max_improv* **do**
    Select one vehicle v_i randomly;
    $j \leftarrow i$;
    if tardiness of $v_i = 0$ **then**
      $j \leftarrow j +$ *step_size*;
    else
      $j \leftarrow j -$ *step_size*;
    end if
    while $j \geq 1$ **and** $j \leq n$ **and** *num_improv* $<$ *max_improv* **do**
      Swap v_i and v_j in S to obtain S';
      if S' is better than S **then**
        $S \leftarrow S'$;
        num_improv $\leftarrow$ *num_improv* $+ 1$;
      end if
      if tardiness of $v_i = 0$ **then**
        $j \leftarrow j +$ *step_size*;
      else
        $j \leftarrow j -$ *step_size*;
      end if
    end while
    chosen_vehicles $\leftarrow$ *chosen_vehicles* $+ 1$;
  end while
  return The current solution S;

The second way to apply the onlooker bee phase (denoted *o_meth*2) chooses EVs to move earlier or to delay depending on the structure of the schedule. Firstly, it considers whether to schedule tardy EVs earlier or to delay EVs without tardiness. This selection is done with probability proportional to the number of tardy EVs. For example, if 20% of EVs have zero tardiness, then we have 80% probability of trying to delay non tardy EVs and only 20% of trying to schedule tardy EVs earlier. The rationale behind this strategy is that when there are many tardy EVs, delaying some non tardy EV gives the chance for a large number of tardy EVs get scheduled earlier—while in situations with a small portion of tardy EVs, it is better trying to schedule earlier one of them each time. Furthermore, an additional control parameter *num_steps* determines the maximum number of swaps that we try for each single EV, so that it helps with further reducing the computational time. The detailed procedure can be seen in Algorithm 3.

5.5. Scout Bee Phase

When a solution cannot be improved after *limit* trials, its food source is abandoned and a scout bee is in charge of looking for a new source. To implement the scout phase, we replace all solutions having *num_trials* $\geq$ *limit* by random ones, and set *num_trials* $= 0$ for each of them.

Algorithm 3 Second type improvement for the onlooker bee phase *o_meth*2

Input A solution S and parameters *step_size*, *max_improv* and *max_steps*
  $probability \leftarrow \#tardy\_EVs/n$;
  $random\_probability \leftarrow$ random number in [0..1];
  $chosen\_vehicles \leftarrow 0$;
  $num\_improv \leftarrow 0$;
  while $chosen\_vehicles < n * 0.1$ **and** $num\_improv < max\_improv$ **do**
    if $random\_probability < probability$ **then**
      Select uniformly one non tardy EV v_i;
      $j \leftarrow i + step\_size$;
    else
      Select uniformly one tardy EV v_i;
      $j \leftarrow i - step\_size$;
    end if
    $num\_steps \leftarrow 0$;
    $improv = False$;
    while $j \geq 1$ **and** $j \leq n$ **and** $num\_steps < max\_steps$ **and not** $improv$ **do**
      Swap v_i and v_j in S to obtain S';
      $num\_steps \leftarrow num\_steps + 1$;
      if S' is better than S **then**
        $S \leftarrow S'$;
        $improv = True$;
        $num\_improv \leftarrow num\_improv + 1$;
      end if
      if $random\_probability < probability$ **then**
        $j \leftarrow j + step\_size$;
      else
        $j \leftarrow j - step\_size$;
      end if
    end while
    $chosen\_vehicles \leftarrow chosen\_vehicles + 1$;
  end while
  return The current solution S;

5.6. Local Search

It is well known that hybridization with local search usually improves the results of an evolutionary metaheuristic, providing extra intensification [46–48]. Here, we propose applying a hill climbing procedure to the final solution reached by a ABC algorithm in the following way: we iterate over the EVs in the order they are scheduled and if the EV in position i (v_i) is tardy, we try to schedule it earlier, just before each one of the $\lfloor i * max\_step\_perc \rfloor$ previous EVs in the schedule, where $max\_step\_perc \in [0, 1]$ is a parameter. If an improving solution is reached, it substitutes the previous one and the iterative process is started again. The local search finishes when no improving solution is reached in a whole iterative process. The parameter *max_step_perc* is given a small value for the sake of efficiency and to introduce reasonably small changes in the neighboring solutions. Notice that the number of EVs for which v_i is swapped depending on the position i; the rationale is that the chances of a EV being scheduled earlier in an improving schedule is in direct ratio with the position in the schedule. Algorithm 4 shows the detailed steps. In Section 6, we will see that this additional local search improves the results of the algorithm, and the extra computational time taken is small due to the fact that it is only applied to the best solution returned by the ABC algorithm.

Algorithm 4 The local search

Input A solution S and the parameter $max\_step\_perc \in [0, 1]$
improvement $\leftarrow$ True;
while improvement **do**
improvement $\leftarrow$ False;
$i \leftarrow 1$;
while $i \leq n$ **do**
if tardiness of $v_i > 0$ **then**
$j = i - 1$;
$k = \lfloor i * max\_step\_perc \rfloor$;
local_improvement $\leftarrow$ False;
while $j \geq i - k$ **and not** local_improvement **do**
Swap v_i and v_j in S to obtain S';
if S' is better than S **then**
$S \leftarrow S'$;
improvement $\leftarrow$ True;
local_improvement $\leftarrow$ True;
end if
$j \leftarrow j - 1$;
end while
end if
$i \leftarrow i + 1$;
end while
end while
return The current solution S;

6. Results

We have considered the benchmark proposed in [8] (publicly available in [49]), which consists of 2160 instances organized in 72 sets of 30 instances each. This benchmark is inspired in a prototype charging station with 180 spaces. The profiles of arrival times, demands and due dates correspond to the expected behavior of real users under different circumstances. The time horizon is one day and three different scenarios were considered: scenario 1 represents a normal week day, where vehicles arrive throughout the day, with two arrival peaks. Scenarios 2 and 3 represent more complex situations where the arrival time of most of the vehicles is at almost the same time. Scenario 3 having tighter due dates than scenario 2. Each set is characterized by the tuple $(scenario, type, N, \Delta)$; $type$ has two possible values 1 and 2, in type 1 instances, one third of the vehicles arrive to each line, whereas in instances of type 2 the loads are unbalanced so that 60% of the vehicles arrive to line 1, 30% to line 2 and 10% to line 3. This unbalance among the lines makes type 2 instances harder to solve due to the difficulty in maintaining the constraint of maximum imbalance. Three values are considered for n, 20, 30 and 40, and four values for Δ, 0.2, 0.4, 0.6 and 0.8.

The proposed algorithm is implemented in C++ programming language using a single thread, and the target machine is Xeon E5520 running Scientific Linux 6.0. Due to the stochastic nature of the ABC algorithm, 30 independent executions were done for each instance to obtain statistically significant results.

For the purpose of comparison, the main reference is the EVS algorithm proposed in [8]. After EVS, some new approaches were proposed, namely a Memetic Algorithm (MA) [9] and an Artificial Bee Colony (ABC) [10], which were able to improve the quality of the solutions from EVS at the cost of taking more execution time. Here, it is important to remark that there are prototype implementations of the charging station, but there are no actual implementations in a real parking yet, and so all previous experimental results considered were obtained by simulations.

To set up the best values of the parameters of the algorithm, we have conducted some experiments using a set of 24 instances, namely the first instance of each group of scenario 1. For all experiments, the stopping condition was adjusted so that the computational time of the different configurations

was similar and comparable to that of other state-of-the-art algorithms. Table 1 summarizes the values tested for each parameter and the best values of the parameters for the static and dynamic versions of the problem.

Table 1. Summary of the values tested for the parameters.

Parameter	Values Tested	Best Static	Best Dynamic
Tournament size (t_{size})	5, 10, 15	10	15
Number of food sources (SN)	100, 300, 500	300	100
Type of employed bee phase	*e_meth*1, *e_meth*2	*e_meth*2	*e_meth*2
Crossover operator	SBX, PMX, At random	At random	At random
Maximum trials for improving a solution (*limit*)	25, 50, 75, 100	50	25
*o_meth*1 :: *step_size*	1, 2, 5, 10	5	2
*o_meth*1 :: *max_improv*	1, 2, 4, 8	2	4
*o_meth*2 :: *step_size*	2, 5, 10	5	2
*o_meth*2 :: *max_steps*	5, 10, 20	20	10
*o_meth*2 :: *max_improv*	2, 4, 8	4	4
Type of onlooker bee phase	*o_meth*1, *o_meth*2	*o_meth*1	*o_meth*1
max_step_perc for the local search	0.05, 0.1, 0.25	0.1	0.1
Final local search step	Apply, Do not apply	Apply	Apply

Using the proposed parameter setting, which is shown in the columns "best static" and "best dynamic" of Table 1, and considering 25 consecutive iterations without improving the best solution obtained so far as stopping condition, we see that the convergence pattern is appropriate. As an example, Figure 2 shows the evolution of the total tardiness in a run of one scenario 1, type 1 instance with $\Delta = 0.2$ and $n = 20$, considering the static version of the problem.

From the results in Table 1, we may draw some interesting conclusions. For example, the second improvement proposed for the employed bee phase (*e_meth*2) performed better than the method *e_meth*1 proposed in [10]. Perhaps this is due to a larger diversity on the generated offspring. In addition, the best configuration for the onlooker bee phase (*o_meth*1 with *step_size* and *max_improv* higher than one) performs better than that proposed in [10] (which uses $step\_size = 1$ and $max\_improv = 1$). It is also worth to mentioning that the final local search step is able to improve the final results of the ABC algorithm when using similar computational times in the comparison.

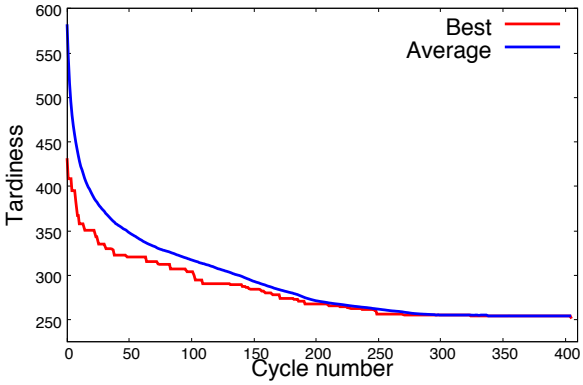


Figure 2. Evolution of the best and average tardiness values along the iterations in one run in an instance of scenario 1, type 1, $\Delta = 0.2$ and $n = 20$, considering the static version of the problem.

Using the best configuration reported in Table 1, we have performed experiments across the full benchmark with 2160 instances. Tables 2–4 report the results in scenarios 1, 2 and 3, respectively. For scenario 1, we report the results obtained by MA [9], ABC [10] and hABC. For the EVS algorithm [8], we only report tardiness values obtained for the dynamic problem, as that paper does not tackle the static

one. For scenarios 2 and 3, we only show tardiness values from hABC along with those reported for EVS [8] and MA [9] on the dynamic problem. As in [10], there are no results reported from ABC on these scenarios. Each value in the Tables represents the sum of the tardiness (in hours) of the 30 instances of each group. The numbers in bold indicate that the best value for each version of the problem.

As it can be expected, the results reported in Tables 2–4 show that the tardiness values strongly depend on the problem characteristics (*scenario*, *type*, n, Δ). Instances with lower n and Δ values present larger tardiness, which seems reasonable as these instances are much more constrained. Type 2 instances have larger tardiness than those of type 1 because of the bottleneck caused by the unequal distribution of EVs in the three lines. In addition, instances of scenario 1 have the lowest tardiness due to the EVs' arrival being evenly distributed along the day, which allows for scheduling many EVs without tardiness. In addition, the tardiness obtained in static problems are generally much lower than those obtained in dynamic ones, which suggests that knowing in advance all the information allows the algorithms to obtain better schedules and that there may still be room to improve the solutions of the dynamic problem.

Table 2. Comparison of hABC to the state-of-the-art methods on the instances of scenario1. The best values obtained for each set of instances are remarked in bold.

n	Δ	Static Problem			Dynamic Problem			
		MA [9]	ABC [10]	hABC	EVS [8]	MA [9]	ABC [10]	hABC
Type 1 instances								
20	0.2	5210.8	5331.7	**5089.0**	8386.3	6981.2	7027.9	**6948.1**
20	0.4	**2509.2**	2586.1	2529.7	4120.4	3884.2	3898.1	**3877.4**
20	0.6	2252.7	2258.2	**2250.6**	3670.6	3556.7	3558.0	**3547.8**
20	0.8	2199.0	2200.4	**2194.1**	3590.9	3503.3	3509.1	**3494.5**
30	0.2	729.5	771.9	**712.6**	1959.3	**1444.8**	1445.7	1445.0
30	0.4	**52.5**	76.8	75.4	421.2	367.6	366.6	**364.8**
30	0.6	**34.3**	34.9	34.4	347.9	**316.2**	317.7	316.8
30	0.8	**33.8**	34.1	33.7	347.6	**314.7**	316.4	315.9
40	0.2	**216.9**	238.6	221.4	735.0	541.6	545.2	**540.3**
40	0.4	**0.0**	**0.0**	**0.0**	14.0	7.7	7.8	**7.4**
40	0.6	**0.0**	**0.0**	**0.0**	**3.4**	**3.4**	**3.4**	**3.4**
40	0.8	**0.0**	**0.0**	**0.0**	**3.4**	**3.4**	**3.4**	**3.4**
Type 2 instances								
20	0.2	**122,615.0**	123,409.0	122,690.0	128,185.0	124,075.0	124,168.0	**123,934.0**
20	0.4	**43991.1**	44,152.7	44,030.0	46,319.3	45,127.2	45,183.9	**45,104.0**
20	0.6	**20,526.9**	20,597.0	20,545.1	22,966.8	21,822.6	21,847.7	**21,788.5**
20	0.8	**12,762.8**	12,734.1	12,727.2	14,573.1	14,209.9	14,212.1	**14,176.8**
30	0.2	**69,610.6**	69,954.3	69,665.3	72,860.8	70,894.5	70,942.6	**70,853.6**
30	0.4	**20,031.7**	20,097.6	20,043.6	21,479.9	21,130.3	21,150.5	**21,109.1**
30	0.6	7030.5	7033.0	**7019.0**	8088.9	7921.5	7923.0	**7905.2**
30	0.8	**3527.9**	3536.4	3530.8	4486.3	4389.3	4391.0	**4384.4**
40	0.2	**43,981.8**	44,146.7	44,024.5	46,135.4	45,139.2	45,192.6	**45,113.7**
40	0.4	9760.7	9775.7	**9753.1**	10,869.3	10,654.6	10,669.2	**10,635.7**
40	0.6	2851.1	2855.7	**2850.4**	3599.1	3515.6	3515.1	**3514.8**
40	0.8	872.3	876.3	**872.0**	1635.5	1572.1	1574.9	**1571.7**

From Tables 2–4, it is also clear that ABC outperforms ABC [10] and EVS [8] in all groups of 30 instances with the same values of parameters n and Δ, in both the static and dynamic versions of the problem. hABC also outperforms MA [9] in most cases, although the differences in the static problem considering scenario 1 are small. In order to assess that the improvement is statistically significant, following [50], we have performed non-parametric statistical tests for cases where we have multiple-problem analysis. First of all, a Shapiro–Wilk test confirmed the non-normality of the data. Then, we used paired Wilcoxon signed rank tests to compare the average results in all instances. In both the static and dynamic versions of the problem, we have obtained p-values lower than 2.2×10^{-16} against the other three state-of-the-art approaches. These tiny p-values confirm that the improvements in these instances are statistically significant, and so we can tell that hABC performs better than the

other methods. If we perform a more detailed analysis, we find that specifically in type 2 instances from scenario 1 in the static version, the difference in the results obtained by hABC and MA is not statistically significant (the *p*-value when assessing if hABC is better than MA is 0.5735, and it is 0.8531 the other way around). In all other cases, hABC is always significantly better.

Table 3. Comparison of hABC to the state-of-the-art methods on the instances of scenario 2. The best values obtained for each set of instances are remarked in bold.

n	Δ	Static Problem		Dynamic Problem		
		MA [9]	hABC	EVS [8]	MA [9]	hABC
Type 1 instances						
20	0.2	14,411.5	**14,097.9**	16,886.0	16,176.3	**15,933.8**
20	0.4	**11,703.6**	11,713.9	14,131.2	13,915.9	**13,869.9**
20	0.6	**11,328.3**	11,335.7	13,648.3	13,559.1	**13,534.8**
20	0.8	**11,242.1**	11,242.9	13,547.9	13,492.7	**13,464.7**
30	0.2	3920.6	**3887.3**	6394.7	5821.5	**5798.8**
30	0.4	2726.2	**2717.1**	4824.3	4688.4	**4687.2**
30	0.6	2592.8	**2588.3**	4668.6	4560.3	**4559.3**
30	0.8	2591.4	**2587.2**	4672.6	4559.0	**4556.7**
40	0.2	**638.0**	662.1	1951.4	1593.5	**1590.0**
40	0.4	**186.4**	**186.4**	1212.7	1102.0	**1101.5**
40	0.6	172.2	**171.5**	1210.7	1103.3	**1102.8**
40	0.8	171.9	**171.6**	1210.6	1102.7	**1102.5**
Type 2 instances						
20	0.2	**137,204.0**	137,271.0	143,883.0	139,192.0	**138,961.0**
20	0.4	57,453.4	**57,295.2**	62,246.2	59,194.0	**59,065.2**
20	0.6	34,548.2	**34,499.5**	39,869.4	36,298.4	**36,231.7**
20	0.8	27,735.1	**27,711.0**	30,887.2	29,791.0	**29,747.5**
30	0.2	82,970.0	**83,224.2**	86,392.0	84,470.0	**84,297.4**
30	0.4	32,714.9	**32,659.2**	34,475.4	33,936.8	**33,891.5**
30	0.6	17,077.3	**17,069.9**	19,355.7	18,709.8	**18,682.5**
30	0.8	12,364.5	**12,359.5**	14,375.1	14,201.7	**14,182.4**
40	0.2	57,652.0	**57,630.0**	59,775.0	58,714.1	**58,591.2**
40	0.4	20,786.9	**20,765.5**	22,254.0	21,984.2	**21,947.1**
40	0.6	9582.4	**9581.7**	10,991.3	10,815.9	**10,808.5**
40	0.8	**5806.0**	5814.1	7260.5	7178.5	**7170.9**

Table 4. Comparison of hABC to the state-of-the-art methods on the instances of scenario 3. The best values obtained for each set of instances are remarked in bold.

n	Δ	Static Problem		Dynamic Problem		
		MA [9]	hABC	EVS [8]	MA [9]	hABC
Type 1 instances						
20	0.2	19,209.6	**18,874.0**	20,704.7	20,242.4	**20,055.1**
20	0.4	16,628.5	**16,632.1**	18,001.7	17,916.8	**17,875.2**
20	0.6	16,218.8	**16,227.3**	**17,528.2**	17,573.4	17,553.6
20	0.8	16,136.2	**16,134.3**	**17,463.3**	17,524.5	17,505.9
30	0.2	7791.9	**7733.0**	9051.1	8569.1	**8556.5**
30	0.4	6528.0	**6524.1**	7347.3	7283.5	**7276.2**
30	0.6	6379.6	**6380.1**	7150.4	7115.3	**7113.5**
30	0.8	6371.6	**6371.2**	7144.5	7106.9	**7103.8**
40	0.2	2422.5	**2438.5**	3478.1	**3106.7**	3108.5
40	0.4	1853.4	**1853.4**	2373.1	**2328.6**	2329.6
40	0.6	1810.3	**1810.0**	2276.7	2247.6	**2247.2**
40	0.8	1810.2	**1810.0**	2276.7	2246.7	**2246.4**

Table 4. *Cont.*

n	Δ	Static Problem		Dynamic Problem		
		MA [9]	hABC	EVS [8]	MA [9]	hABC
Type 2 instances						
20	0.2	129,339.0	**129,171.0**	138,064.0	131,037.0	**130,903.0**
20	0.4	56,672.0	**56,513.3**	62,988.8	58,225.8	**58,080.9**
20	0.6	37,089.8	**37,034.4**	42,528.3	38,574.6	**38,507.2**
20	0.8	31,715.6	**31,705.2**	33,998.2	33,360.3	**33,313.8**
30	0.2	**79,182.4**	79,359.1	83,169.5	80,526.7	**80,393.4**
30	0.4	33,084.2	**33,021.5**	34,799.7	34,045.1	**33,990.5**
30	0.6	19,592.6	**19,559.8**	21,321.1	20,645.1	**20,622.7**
30	0.8	15,812.5	**15,805.1**	16,972.0	16,979.7	**16,962.8**
40	0.2	55,832.1	**55,747.9**	58,306.3	56,737.6	**56,635.1**
40	0.4	21,932.0	**21,901.8**	22,988.7	22,710.2	**22,680.2**
40	0.6	11,462.9	**11,455.3**	12,220.3	12,168.1	**12,161.5**
40	0.8	8388.2	**8387.9**	**9127.3**	9204.6	9198.8

In order to visualize the structure of the schedules and to appreciate how the difficulty of the problems strongly depend on the different loads of the three lines, we show two complete schedules (types 1 and 2 respectively, $n = 30$ and $\Delta = 0.4$ in both cases) for the dynamic problem in Figures 3 and 4 (more schedulesare included in the complementary material of the paper). The schedule in Figure 3 shows a normal situation where the load is balanced in the three lines so that all EVs are scheduled almost without tardiness and at the same time the maximum imbalance ($\Delta = 0.4$) is only reached over small periods of time. However, the schedule in Figure 4 represents an extreme situation (which probably will not happen in the real environment) where EVs are unevenly distributed on the lines. As a consequence, the imbalance is at a maximum over almost the entire scheduling horizon and the tardiness in line 1 is very high due to the low load in lines 2 and 3. For this reason, the charge of EVs in line 1 is delayed up to one day more than in the schedule of Figure 3.

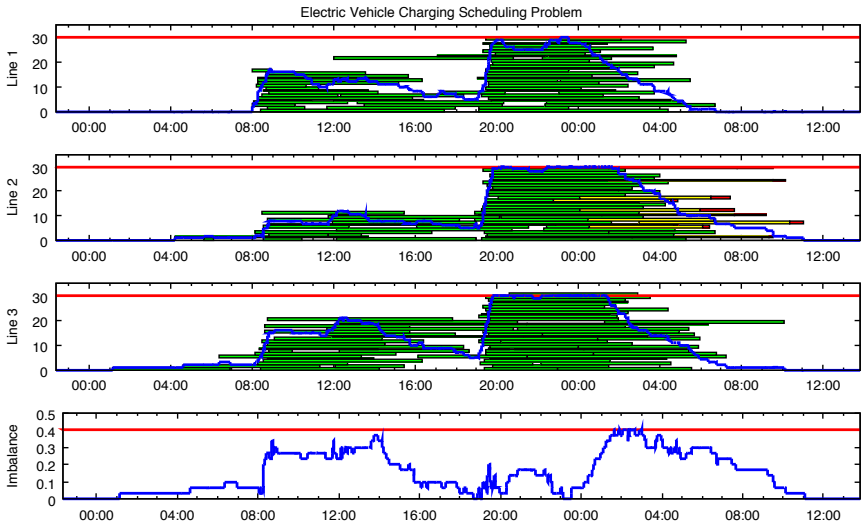


Figure 3. Example of a feasible schedule of a Type 1 instance of scenario 1, with $n = 30$ and $\Delta = 0.4$, showing in green the charging intervals of the EVs without tardiness, and in yellow and red the EVs that complete charging after their due dates. The red portion indicates the interval after the corresponding due date. Blue plots in each of the three lines show the load level $N_i(t)$, which is always lower than or equal to $n = 30$. The bottom graph shows the imbalance at each time point in the three lines, which is always lower than or equal to $\Delta = 0.4$.

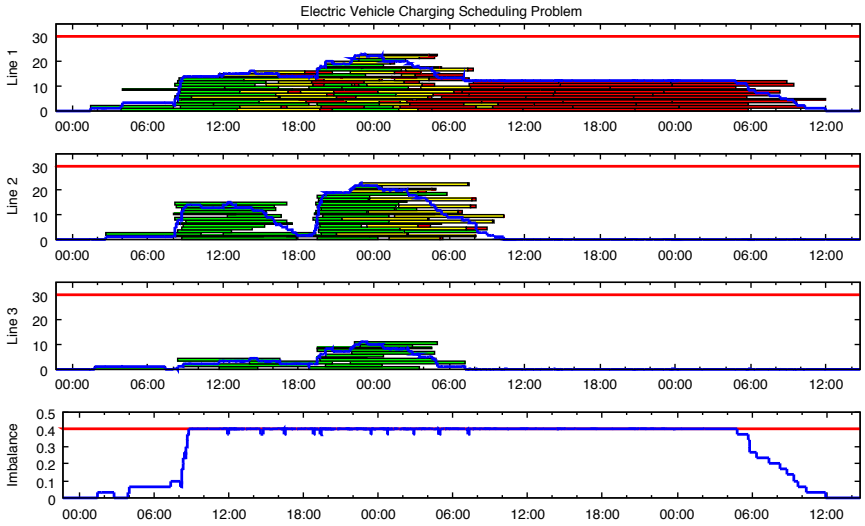


Figure 4. Example of a feasible schedule of a Type 2 instance of scenario 1, with $n = 30$ and $\Delta = 0.4$.

The EVs charging scheduling algorithm must be run in a real-time setting, and so it is quite relevant to consider its computational cost. Thus, we should check if hABC requires much lower time than 120 s to solve any instance of the dynamic problem, as this is the time lapse between consecutive instances (see Section 4.2). We have seen that the instances requiring the most computational time are those of type 2, $n = 20$ and $\Delta = 0.2$ (see Table 5). The average time required by hABC to solve an instance having those parameters was 1.88 s, being 12.34 s in the worst case. Clearly, these times are much lower than two minutes and hence we can conclude that hABC can be used in the real setting. The other metaheuristic methods require similar computational times: MA [9] took an average of 1.23 s per instance and 10.26 s in the worst case, and ABC [10] required 1.97 s in average and 15.03 s in the worst case. On the other hand, EVS [8] is the fastest algorithm, taking less than 0.012 s in all runs, although it also obtained the worst results; this is reasonable due to using simple dispatching rules.

Regarding the static version of the problem, the computational time required by hABC is also comparable to that of the other metaheuristic approaches. In Table 5, we report the average running times for scenario 1 instances, depending on the instance type and the parameters n and Δ, in both the static and dynamic versions of the problem. We have to remark that the values of the dynamic problem represent the total execution time for solving all instances (up to 720 in the 24-h period) each one with the EVs in the station that have not started to charge at the scheduling point, whereas in the static problem there is only one large instance with all EVs over the whole time horizon. It is also worth to noting that, in the dynamic problem, all EVs that have not yet started to charge have to be rescheduled, and so the same vehicle may appear in several consecutive instances, which also justifies larger values than those of the static version, particularly in type 2 instances. For all the above, the comparison of the times required to solve the static and dynamic problems lacks real significance. What is really relevant is the gap between their tardiness, which provides some hints about how much the solution of the dynamic problem could be further improved with powerful algorithms or just by having some knowledge on EV arrival.

Table 5. Comparison of average execution times (in seconds) with the state of the art in scenario 1 instances. The values of the dynamic problem represent the total execution time for solving all instances (up to 720 in the 24-h period).

n	Δ	Static Problem			Dynamic Problem		
		MA [9]	ABC [10]	hABC	MA [9]	ABC [10]	hABC
Type 1 instances							
20	0.2	75.9	111.0	102.6	144.3	136.5	113.4
20	0.4	50.9	57.2	62.4	65.7	63.5	48.6
20	0.6	39.2	43.7	45.0	48.2	48.5	36.1
20	0.8	37.2	41.5	42.8	43.9	45.1	37.5
30	0.2	31.2	47.1	50.0	23.0	28.7	9.5
30	0.4	6.5	11.1	9.4	8.7	12.1	2.7
30	0.6	3.5	7.4	4.6	7.8	10.7	2.2
30	0.8	3.4	7.4	4.4	7.7	10.7	2.2
40	0.2	11.4	18.3	19.9	10.6	13.8	4.0
40	0.4	0.3	0.2	0.2	0.5	0.9	0.2
40	0.6	0.2	0.2	0.2	0.1	0.2	0.1
40	0.8	0.2	0.2	0.2	0.1	0.2	0.1
Averages		21.7	28.8	28.5	30.1	30.9	21.4
Type 2 instances							
20	0.2	115.3	271.9	166.2	437.6	1512.2	860.1
20	0.4	101.3	201.5	141.3	236.8	684.1	450.5
20	0.6	89.6	150.9	115.8	200.6	346.7	311.0
20	0.8	78.1	114.4	90.4	161.6	223.2	212.3
30	0.2	107.9	233.2	151.4	311.4	1002.7	608.7
30	0.4	87.6	143.5	109.0	164.2	298.9	223.9
30	0.6	61.7	89.6	75.3	95.3	114.4	85.7
30	0.8	46.6	57.8	55.0	59.5	63.3	43.2
40	0.2	101.6	205.4	140.6	233.2	682.7	439.9
40	0.4	71.9	104.6	85.2	115.1	149.8	115.6
40	0.6	41.9	53.6	51.0	49.3	52.4	32.5
40	0.8	26.8	41.5	28.7	26.6	27.3	10.9
Averages		77.5	139.0	100.8	174.3	429.8	282.9

7. Conclusions

In this paper, we have proposed a new algorithm to solve the electric vehicle charging scheduling problem proposed in [8]. This algorithm combines artificial bee colony with local search algorithms and other improvements. By experimental study, we have analyzed our proposal and compared it with the state of the art on a set of real-world inspired instances. The proper balance between diversification and intensification allows this hybrid algorithm to obtain very good results on these instances. Moreover, the reasonable computational time taken by our algorithm allows it to be used in the real environment for online scheduling.

The main contributions of the paper may be summarized as follows:

- We have devised some strategies to introduce domain knowledge in the artificial bee colony algorithm proposed in [10], namely a neighborhood structure which was exploited in local search and some improvements in the employer and onlooker bee phases.
- The new hybrid algorithm outperforms all previous methods in terms of tardiness penalties. Therefore, it contributes to make the best use of the contracted power in the charging station and to satisfy the users' demands at the same time.
- From the experimental study on the static and dynamic versions of the electric vehicle charging scheduling problem, we have gained interesting insights into the problem structure, which

will allow for incorporating new features and objective functions in future extensions of the charging model.

As future work, we will consider more realistic charging models, as those mentioned in Section 4.3, and incorporate new objective functions as minimizing peak consumption or the imbalance between the lines, which will require modeling and solving the problem in the frameworks of multi or many objective optimization.

Author Contributions: The four authors contributed to elaborate the paper. J.G.Á. implemented the algorithms and performed the experimental study. M.Á.G. designed the improvements introduced in the artificial bee colony algorithm and wrote the paper. C.R.V. supervised the research work and made a critical review of the paper. R.V. provided the conceptualization of the problem, gave feedback in all steps of the research work and made a critical review of the paper.

Funding: This research was funded by Spanish Government grant TIN2016-79190-R.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ABC	Artificial Bee Colony
DDR	Due Date Rule
EV	Electric Vehicle
EVS	Electric Vehicle Scheduling
GRASP	Greedy Randomized Adaptive Search Procedure
hABC	Hybrid Artificial Bee Colony
LST	Latest Starting Time
MA	Memetic Algorithm
PMX	Partially-Mapped Crossover
PSO	Particle Swarm Optimization
SBX	Starting-time Based Crossover

References

1. Awasthi, A.; Venkitusamy, K.; Padmanaban, S.; Selvamuthukumaran, R.; Singh, A.K. Optimal planning of electric vehicle charging station at the distribution system using hybrid optimization algorithm. *Energy* **2017**, *133*, 70–78. [CrossRef]
2. Guo, C.; Yang, J.; Yang, L. Planning of Electric Vehicle Charging Infrastructure for Urban Areas with Tight Land Supply. *Energies* **2018**, *11*, 2314. [CrossRef]
3. Kong, C.; Jovanovic, R.; Bayram, I.S.; Devetsikiotis, M. A Hierarchical Optimization Model for a Network of Electric Vehicle Charging Stations. *Energies* **2017**, *10*, 675. [CrossRef]
4. Chen, Y.W.; Cheng, C.Y.; Li, S.F.; Yu, C.H. Location optimization for multiple types of charging stations for electric scooters. *Appl. Soft Comput.* **2018**, *67*, 519–528. [CrossRef]
5. Lim, Y.; Kim, H.M.; Kang, S.; Kim, T.H. Vehicle-to-grid communication system for electric vehicle charging. *Integr. Comput. Aided Eng.* **2012**, *19*, 57–65. [CrossRef]
6. García-Villalobos, J.; Zamora, I.; San Martín, J.; Asensio, F.; Aperribay, V. Plug-in Electric Vehicles in Electric Distribution Networks: A Review of Smart Charging Approaches. *Renew. Sustain. Energy Rev.* **2014**, *38*, 717–731. [CrossRef]
7. Rahman, I.; Vasant, P.; Singh, B.; Abdullah-Al-Wadud, M.; Adnan, N. Review of recent trends in optimization techniques for plug-in hybrid, and electric vehicle charging infrastructures. *Renew. Sustain. Energy Rev.* **2016**, *58*, 1039–1047. [CrossRef]
8. Hernandez-Arauzo, A.; Puente, J.; Varela, R.; Sedano, J. Electric Vehicle Charging under Power and Balance Constraints as Dynamic Scheduling. *Comput. Ind. Eng.* **2015**, *85*, 306–315. [CrossRef]
9. García-Álvarez, J.; González, M.; Vela, C. Metaheuristics for solving a real-world electric vehicle charging scheduling problem. *Appl. Soft Comput.* **2018**, *65*, 292–306. [CrossRef]

10. García-Álvarez, J.; González, M.; Vela, C.; Varela, R. Electric Vehicle Charging Scheduling Using an Artificial Bee Colony Algorithm. In *Lecture Notes in Computer Science, Proceeding of the International Work-Conference on the Interplay between Natural and Artificial Computation for Biomedicine and Neuroscience IWINAC 2017, Corunna, Spain, 19–23 June 2017*; Ferrández Vicente, J., Álvarez-Sánchez, J., de la Paz López, F., Toledo Moreo, J., Adeli, H., Eds.; Springer: Cham, Switzerland, 2017; Volume 10337, pp. 115–124. [CrossRef]
11. Rahman, I.; Vasant, P.M.; Singh, B.S.M.; Abdullah-Al-Wadud, M. Novel metaheuristic optimization strategies for plug-in hybrid electric vehicles: A holistic review. *Intell. Decis. Technol.* **2016**, *10*, 149–163. [CrossRef]
12. Xu, S.; Feng, D.; Yan, Z.; Zhang, L.; Li, N.; Jing, L.; Wang, J. Ant-based swarm algorithm for charging coordination of electric vehicles. *Int. J. Dis. Sens. Netw.* **2013**, *2013*, 13. [CrossRef]
13. Karfopoulos, E.; Hatziargyriou, N. Distributed coordination of electric vehicles for conforming to an energy schedule. *Electr. Power Syst. Res.* **2017**, *151*, 86–95. [CrossRef]
14. Zhang, W.; Zhang, D.; Mu, B.; Wang, L.Y.; Bao, Y.; Jiang, J.; Morais, H. Decentralized Electric Vehicle Charging Strategies for Reduced Load Variation and Guaranteed Charge Completion in Regional Distribution Grids. *Energies* **2017**, *10*, 147. [CrossRef]
15. Hu, Z.; Zhan, K.; Zhang, H.; Song, Y. Pricing mechanisms design for guiding electric vehicle charging to fill load valley. *Appl. Energy* **2016**, *178*, 155–163. [CrossRef]
16. Langbroek, J.H.; Franklin, J.P.; Susilo, Y.O. When do you charge your electric vehicle? A stated adaptation approach. *Energy Policy* **2017**, *108*, 565–573. [CrossRef]
17. Ma, Z.; Callaway, D.; Hiskens, I. Decentralized Charging Control of Large Populations of Plug-in Electric Vehicles. *IEEE Trans. Control Syst. Technol.* **2013**, *21*, 67–78. [CrossRef]
18. Foster, J.; Caramanis, M. Optimal power market participation of plug-in electric vehicles pooled by distribution feeder. *IEEE Trans. Power Syst.* **2013**, *28*, 2065–2076. [CrossRef]
19. Jian, L.; Zheng, Y.; Shao, Z. High efficient valley-filling strategy for centralized coordinated charging of large-scale electric vehicles. *Appl. Energy* **2017**, *186*, 46–55. [CrossRef]
20. He, Y.; Venkatesh, B.; Guan, L. Optimal Scheduling for Charging and Discharging of Electric Vehicles. *IEEE Trans. Smart Grid* **2012**, *3*, 1095–1105. [CrossRef]
21. Wu, H.; Pang, G.K.H.; Choy, K.L.; Lam, H.Y. Dynamic resource allocation for parking lot electric vehicle recharging using heuristic fuzzy particle swarm optimization algorithm. *Appl. Soft Comput.* **2018**, *71*, 538–552. [CrossRef]
22. García-Álvarez, J.; González, M.; Vela, C. A Genetic Algorithm for Scheduling Electric Vehicle Charging. In Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, Madrid, Spain, 11–15 July 2015; pp. 393–400.
23. Yoon, S.G.; Kang, S.G. Economic Microgrid Planning Algorithm with Electric Vehicle Charging Demands. *Energies* **2017**, *10*, 1487. [CrossRef]
24. Gan, L.; Topcu, U.; Low, S. Optimal decentralized protocol for electric vehicle charging. *IEEE Trans. Power Syst.* **2013**, *28*, 940–951. [CrossRef]
25. Lopes, J.; Soares, F.; Almeida, P.; Moreira da Silva, M. Smart charging strategies for electric vehicles: Enhancing grid performance and maximizing the use of variable renewable energy resources. In Proceedings of the EVS24 International Battery, Hybrid and Fuel Cell Electric Vehicle Symposium, Stavanger, Norway, 13–16 May 2009; pp. 392–396.
26. Kang, J.; Duncan, S.J.; Mavris, D.N. Real-time Scheduling Techniques for Electric Vehicle Charging in Support of Frequency Regulation. *Procedia Comput. Sci.* **2013**, *16*, 767–775. [CrossRef]
27. Janjic, A.; Velimirovic, L.; Stankovic, M.; Petrusic, A. Commercial electric vehicle fleet scheduling for secondary frequency control. *Electr. Power Syst. Res.* **2017**, *147*, 31–41. [CrossRef]
28. Han, J.; Park, J.; Lee, K. Optimal Scheduling for Electric Vehicle Charging under Variable Maximum Charging Power. *Energies* **2017**, *10*, 933. [CrossRef]
29. Ma, C.; Rautiainen, J.; Dahlhaus, D.; Lakshman, A.; Toebermann, J.C.; Braun, M. Online optimal charging strategy for electric vehicles. *Energy Procedia* **2015**, *73*, 173–181. [CrossRef]
30. Su, W.; Chow, M.Y. Performance evaluation of an EDA-based large-scale plug-in hybrid electric vehicle charging algorithm. *IEEE Trans. Smart Grid* **2012**, *3*, 308–315. [CrossRef]
31. Iversen, E.; Morales, J.; Madsen, H. Optimal charging of an electric vehicle using a Markov decision process. *Appl. Energy* **2014**, *123*, 1–12. [CrossRef]

32. Wu, F.; Sioshansi, R. A two-stage stochastic optimization model for scheduling electric vehicle charging loads to relieve distribution-system constraints. *Trans. Res. Part B Methodol.* **2017**, *102*, 55–82. [CrossRef]
33. Tran, T.; Dogru, M.; Ozen, U.; Beck, C. Scheduling a multi-cable electric vehicle charging facility. In Proceedings of the SPARK'13, ICAPS'13 Scheduling and Planning Applications woRKshop, ICAPS Council, Rome, Italy, 10–14 June 2013; pp. 20–26.
34. Mumtaz, S.; Ali, S.; Ahmad, S.; Khan, L.; Hassan, S.Z.; Kamal, T. Energy Management and Control of Plug-In Hybrid Electric Vehicle Charging Stations in a Grid-Connected Hybrid Power System. *Energies* **2017**, *10*, 1923. [CrossRef]
35. Umetani, S.; Fukushima, Y.; Morita, H. A linear programming based heuristic algorithm for charge and discharge scheduling of electric vehicles in a building energy management system. *Omega* **2017**, *67*, 115–122. [CrossRef]
36. Yang, J.; He, L.; Fu, S. An improved PSO-based charging strategy of electric vehicles in electrical distribution grid. *Appl. Energy* **2014**, *128*, 82–92. [CrossRef]
37. Xydas, E.; Marmaras, C.; Cipcigan, L.M. A multi-agent based scheduling algorithm for adaptive electric vehicles charging. *Appl. Energy* **2016**, *177*, 354–365. [CrossRef]
38. Arias, N.B.N.; Franco, J.F.; Lavorato, M.; Romero, R. Metaheuristic optimization algorithms for the optimal coordination of plug-in electric vehicle charging in distribution systems with distributed generation. *Electr. Power Syst. Res.* **2017**, *142*, 351–361. [CrossRef]
39. Sedano, J.; Portal, M.; Hernández-Arauzo, A.; Villar, J.R.; Puente, J.; Varela, R. Intelligent System for Electric Vehicle Charging: Design and Operation. *DYNA* **2013**, *88*, 644–651. [CrossRef]
40. IEC. *Electric Vehicle Conductive Charging System?Part 1: General Requirement*; IEC 61851-1; IEC: London, UK, 2001.
41. Alonso, M.; Amaris, H.; Germain, J.G.; Galan, J.M. Optimal Charging Scheduling of Electric Vehicles in Smart Grids by Heuristic Algorithms. *Energies* **2014**, *7*, 2449–2475. [CrossRef]
42. Karaboga, D. *An Idea Based on Honey Bee Swarm for Numerical Optimization*; Technical Report-tr06; Erciyes University, Engineering Faculty, Computer Engineering Department: Kayseri, Turkey, 2005.
43. Karaboga, D.; Gorkemli, B.; Ozturk, C.; Karaboga, N. A comprehensive survey: artificial bee colony (ABC) algorithm and applications. *Artifi. Intell. Rev.* **2014**, *42*, 21–57. [CrossRef]
44. Pan, Q.K.; Fatih Tasgetiren, M.; Suganthan, P.N.; Chua, T.J. A Discrete Artificial Bee Colony Algorithm for the Lot-streaming Flow Shop Scheduling Problem. *Inf. Sci.* **2011**, *181*, 2455–2468. [CrossRef]
45. Goldberg, D.E.; Lingle, R. Alleles, Loci, and the Traveling Salesman Problem. In *Proceedings of the First International Conference on Genetic Algorithms and Their Application*; Lawrence Erlbaum: Hillsdale, NJ, USA, 1985; pp. 154–159.
46. Gao, J.; Sun, L.; Gen, M. A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. *Comput. Oper. Res.* **2008**, *35*, 2892–2907. [CrossRef]
47. Puente, J.; Vela, C.R.; González-Rodríguez, I. Fast Local Search for Fuzzy Job Shop Scheduling. In *Frontiers in Artificial Intelligence and Applications*; IOS Press: Amsterdam, The Netherlands, 2010; pp. 739–744.
48. Vela, C.R.; Varela, R.; González, M.A. Local Search and Genetic Algorithm for the Job Shop Scheduling Problem with Sequence Dependent Setup Times. *J. Heuristics* **2010**, *16*, 139–165. [CrossRef]
49. Repository. Available online: http://www.di.uniovi.es/iscop (accessed on 2 September 2018).
50. García, S.; Fernández, A.; Luengo, J.; Herrera, F. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental Analysis of Power. *Inf. Sci.* **2010**, *180*, 2044–2064. [CrossRef]

Capítulo 9

Publicaciones en congresos relevantes según el GII/GRIN/SCIE Conference Rating

En este capítulo incluimos las publicaciones presentadas en congresos relevantes según el GII/GRIN/SCIE Conference Rating.

9.1. A genetic algorithm for scheduling electric vehicle charging

Presentamos la siguiente publicación:

- **Título:** A genetic algorithm for scheduling electric vehicle charging.
- **Congreso:** *Genetic and Evolutionary Computation Conference (GECCO 2015).*
- **Año:** 2015.
- **Referencia:** García-Álvarez, J.; González, M.A.; Vela, C.R. A Genetic Algorithm for Scheduling Electric Vehicle Charging. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2015).* Madrid, July 2015, 393–400. DOI: 10.1145/2739480.2754695.

A Genetic Algorithm for Scheduling Electric Vehicle Charging

Jorge García-Álvarez
Dept. of Computer Science
University of Oviedo
Gijón, Spain
jgarcia-alvarez@outlook.com

Miguel A. González
Dept. of Computer Science
University of Oviedo
Gijón, Spain
mig@uniovi.es

Camino R. Vela
Dept. of Computer Science
University of Oviedo
Gijón, Spain
crvela@uniovi.es

ABSTRACT

This paper addresses a problem motivated by a real life environment, in which we have to schedule the charge of electric vehicles in a parking, subject to a set of constraints, with the objective of minimizing the total tardiness. We consider both the static version of the problem, where we know in advance the arrival time, charging time and due date of every vehicle, and also the dynamic version of it. We design a genetic algorithm with some components specifically tailored to deal with the problem. In the experimental study we evaluate the proposed algorithm in a benchmark set taken from the literature, and we also compare it against the state-of-the-art showing that our proposal is significantly better.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods, Scheduling*

Keywords

Time-tabling and scheduling; Genetic algorithms; Heuristics; Combinatorial optimization; Dynamical optimization

1. INTRODUCTION

The number of electric vehicles (EVs) has increased in last years. They could have a very positive impact on the economies of the countries and also in the environment, reducing the dependence on oil. In addition, the energy stored in electric vehicles can be used as an auxiliary source of electrical energy storage [8] to regulate voltage profiles and power of the network, and compensate for fluctuations in renewable energy generation [1]. In this way, EVs can contribute to the efficient use of energy. However, the use of large fleets of EVs may overload the grid due to the high charging time batteries. As it is noted in the report of the European Distribution System Operators for Smart Grids [2], one of the challenges in electric vehicle technology is the development of intelligent control systems for charging that avoid peak demand. The charging of vehicles can be scheduled to take place at times of low consumption and thus balance the electrical needs when vehicles are parked for long periods of time. In fact, a number of charging control systems have already been proposed [3] [10] that, in some cases, try to fill the overnight valley in order to reduce daily cycling and operational cost.

In this paper we face the problem presented and defined in [6], which consists in scheduling the charging of a large number of electric vehicles so that the user demands are met. Calculating a schedule that maximizes user satisfaction can be a very difficult problem to solve due to the physical constraints of charging stations and the existence of contracts with electrical companies that provide a limited amount of energy. The origin of this problem is inspired by a real project in which users require the charge of their electric vehicles while they are parked in their private parking spaces. Each parking space has a single connection to a three-phase line that has certain restrictions. Not all EVs can be charged simultaneously because the power supply is limited and must be balanced between the three lines, thus a suitable control policy is needed to define the intervals for charging vehicles during the time they are stationed [11]. In [6] the authors define this problem and propose an algorithm for solving its dynamic version. Their Problem Decomposition (PD) approach is based on decomposing the problem in three instances of the one machine sequencing problem with variable capacity, and these instances are solved with a method based on a dispatching rule.

However, the problem has a great complexity, and even if heuristic schedulers are able to obtain reasonable solutions, metaheuristics are better suited for obtaining good results. In fact, a number of metaheuristic approaches can be found in the literature to solve similar problems, as for example particle swarm optimization [12]. In [4] we can find a review of different strategies, algorithms and methods to implement smart charging control systems.

In particular, genetic algorithms (GAs) have the ability to work with huge search spaces and allow to exploit any kind of heuristic knowledge from the problem domain, and by doing so, they are actually competitive with the most efficient methods in scheduling. In this paper we present a GA with operators designed specifically for this problem. We propose an scheduler algorithm which incorporates a procedure to fix some inefficient charging time assignments that may appear due to the constraint of maximum imbalance

GECCO '15, July 11–15, 2015, Madrid, Spain

 ISBN 978-1-4503-3472-3/15/07...$15.00
DOI: http://dx.doi.org/10.1145/2739480.2754695

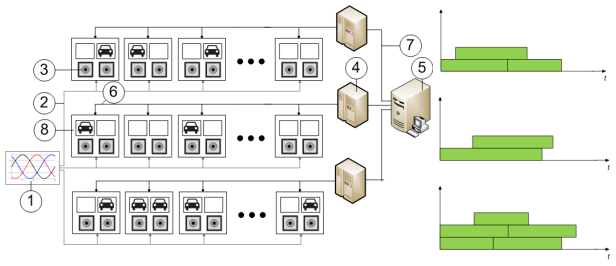


Figure 1: General structure of the distribution net of the charging station. (1) power source, (2) three-phase electric power, (3) charging points, (4) masters, (5) server with database, (6) communication RS 485, (7) communication TCP/IP, (8) slaves. The Gantt Chart in the right side shows the charging schedule of the vehicles in each line.

between any two lines. We also propose some crossover operators tailored for this problem and we prove that one of them obtains significantly better results than standard operators. Additionally, we define a mutation operator for this problem and a way of generating a diverse initial population.

We are considering two versions of the problem: the static version in which we know in advance the arrival times, charging times and due dates of all the vehicles, and the dynamic version in which we do not have that information. We will see that both versions of the problem are relevant. We conducted an experimental study to analyze our proposal and to compare it against the state-of-the-art, and also to discuss the difference in solving the static and dynamic problems.

The remainder of the paper is organized as follows. In the next section we summarize the aspects of the charging station. Then, we describe the problem and the algorithm to solve it. Finally, we report the results of the experimental study and give conclusions and ideas for future research.

2. THE CHARGING STATION

In this section we summarize the main features of the electrical structure and the operation mode of the charging station. These elements are detailed in [11]. Figure 1 shows a schema of the distribution net of the charging station. We have to remark that we had no choice over its design, which was motivated by simplicity and economic reasons. The net is feeded by a three-phase source of electric power. In the model considered here, each line feeds a number of charging points. The station has 180 spaces (60 in each line), each one having a charging point which may be in two states: active or inactive. When it is active it is connected to the net and transfers energy at a constant rate (2.3 Kw).

The operation of the station is controlled by a distributed system comprising one master in each line and a number of slaves. Each user has one space assigned. It is important to remark that the user has to be the owner or renter of its stall and he cannot use the space of another user. This restriction makes the scheduling problem harder to solve as each stall is connected to one of the three lines of the three phase feeder and so keeping the balance constraints may not be easy. When entering the station, the user connects his vehicle to the charging point and provides the charging time, as well as an expected time, or due date, for taking the vehicle away. These values are used by the control system to schedule the vehicle, i.e., to establish a starting time.

There are some constraints that must be satisfied for the station to work properly. For example, although there are 180 spaces available, not all the charging points in these spaces can be activated at the same time. In practice there is a maximum number of vehicles N, which depends on the contracted power, that can be charging (active) at the same time in a given line. Also, due to electro-technical and economical reasons, the power consumption in the three lines must be balanced. This condition is considered as a maximum imbalance between any two lines.

In this paper we consider a simplified model of the charging station that makes the following assumptions: the user never takes the vehicle away before the declared due date and the battery does not get completely charged before the charging time indicated by the user. Even though they are unrealistic assumptions, the model could be easily adapted to deal with these situations.

In principle, each time a new vehicle enters the facility and requires charging, the current schedule may get unfeasible and so a new schedule should be built. However, in order to avoid the system to collapse if many of such events are produced in a very short period of time, new schedules are computed at most at time intervals of length ΔT. To this end, every ΔT time units a supervisor program, running on the server, checks for the events produced in the last interval. If at least one event was produced, then the scheduler is launched to obtain a new feasible schedule which is applied from this time on. In the real environment, ΔT is set at two minutes.

3. PROBLEM FORMULATION

The problem defined in [6] may naturally be considered as a dynamic problem due to the fact that in a real scenario, the arrival of the vehicles is not known in advance. However, in this paper we will consider both static and dynamic versions as it is interesting to compare the difficulty of solving each version. Moreover, the study of the static version of the problem can give some insights about how to solve the dynamic version more efficiently and also gives information about how much the solution to the dynamic problem can still be improved. We borrow the problem formulation from [6], which is detailed in the following subsections.

3.1 The static problem

Considering that we know in advance the arrival times of the vehicles, their charging times and due dates, we can formalize the static version of the problem.

In an instance of the static version of the problem, there are three lines $L_i, 1 \leq i \leq 3$, each one with n_i charging points. $N > 0$ is the maximum number of charging points that can be active at the same time in each line. The line L_i receives a number of M_i vehicles $v_{i1}, \ldots, v_{iM_i}$ from a time 0 up to a planning horizon. Each vehicle v_{ij} is characterized by an arrival time $t_{ij} \geq 0$, a charging time $p_{ij} > 0$ and a time at which the user is expected to take the vehicle away, or due date, $d_{ij} \geq t_{ij} + p_{ij}$ by which the battery of the vehicle should be completely charged.

The goal is to obtain a feasible schedule, i.e. to assign starting times to the decision variables st_{ij} for each v_{ij}, $1 \leq i \leq 3$, $1 \leq j \leq M_i$, so that the following constraints are satisfied and the evaluation function is minimized:

1. All vehicles must start charging after the arrival time.

$$\forall v_{ij}, \quad st_{ij} \geq t_{ij} \tag{1}$$

2. No preemption is allowed, so a vehicle v_{ij} can not be disconnected before its charging time C_{ij} is reached.

$$C_{ij} = st_{ij} + p_{ij} \tag{2}$$

3. The number of active charging points in a line at a given time can not exceed N.

$$\max N_i(t) \leq N, \quad t \geq 0; 1 \leq i \leq 3 \tag{3}$$

where $N_i(t)$ denotes the number of charging points of line L_i which are active at a time t.

4. The maximum imbalance between any two lines L_i and L_j is controlled by the parameter Δ.

$$\max \left(\frac{|N_i(t) - N_j(t)|}{N} \right) \leq \Delta, \quad t \geq 0; 1 \leq i, j \leq 3 \tag{4}$$

The objective function that should be minimized is the total tardiness, defined as

$$\sum_{i=1}^{3} \sum_{j=1}^{M_i} \max(0, C_{ij} - d_{ij}) \tag{5}$$

3.2 The dynamic problem

An instance of the dynamic problem can be given as a sequence of instances $P_1, P_2, \ldots, P_n$. Each P_k is defined by a set of vehicles that demand charging at a given time, but have not yet started to charge, along with some vehicles that may have already started to charge but are still charging.

In a more formal definition, we can consider that a set of vehicles $\{v_{i1}, \ldots, v_{ia_i}, \ldots, v_{im_i}\}$ are given in an instance P_k at a time T_k in each line L_i, $1 \leq i \leq 3$. Each vehicle v_{ij} has a charging time p_{ij} and has a due date d_{ij}. The vehicles $v_{i1}, \ldots, v_{ia_i}$ are already active, as they started to charge at a time $st_{ij} < T_k$, $1 \leq j \leq a_i$ and have not yet finished, i.e., $C_{ij} = st_{ij} + p_{ij} > T_k$. While the vehicles $v_{ia_i+1}, \ldots, v_{im_i}$ have not yet started to charge. So, in the iteration k, the capacity of the line L_i to charge new vehicles, denoted $M_i^k(t)$ is given by

$$M_i^k(t) = N - \sum_{j=1}^{a_i} X_{ij}(t), \quad t \geq T_k \tag{6}$$

where

$$X_{ij}(t) = \begin{cases} 1, & \text{if } t < C_{ij} \\ 0, & \text{if } t \geq C_{ij} \end{cases} \tag{7}$$

The goal of an instance P_k is to obtain a feasible schedule such that all vehicles can be charged, even if no new vehicles arrive after T_k. This requires assigning starting times st_{ij}^* to all vehicles unscheduled at time T_k so that all constraints naturally derived from the static problem are satisfied. Notice that a given vehicle may get different values for st_{ij}^* in subsequent iterations while it does not start charging. However, once it starts, it can not longer be rescheduled. The evaluation function to minimize is again the total tardiness.

4. SOLVING THE DYNAMIC PROBLEM

Algorithm 1 follows the approach presented in [6] and shows the procedure to solve the dynamic version of the problem, where ΔT represents the length of the time interval at which a new event may be expected (two minutes in the real setting). The algorithm iterates every ΔT, so in each T_k, a new instance P_k is created if any vehicle has arrived since T_{k-1}. This instance is solved by the genetic algorithm and the new solution S' replaces the current one from T_k on. If no vehicle has arrived between T_{k-1} and T_k then the current solution remains valid until the next iteration. This means that the vehicles start charging at the times st_{ij}^* given in the current solution S', and therefore $st_{ij} = st_{ij}^*$ and $C_{ij} = st_{ij} + p_{ij}$ in the final solution S.

Algorithm 1 Solving the dynamic version of the problem

Require: A problem instance P and the time interval ΔT
  $S \leftarrow \varnothing$; $S' \leftarrow \varnothing$; $k = 1$; $T_{k-1} = 0$; $T_k = T_{k-1} + \Delta T$;
  while Not all vehicles have been scheduled **do**
    if A new vehicle v_{ij} has arrived at a time $t = t_{ij} \in [T_{k-1}, T_k)$ **then**
      Generate a new instance P_k with all vehicles v_{ij} such that $t_{ij} < T_k$ and that have not yet started charging along with all vehicles still charging at time T_k;
      Calculate a solution S' for instance P_k; {i.e. define starting times $st_{ij}^* \geq T_k$ for the vehicles of P_k that are not yet charging}
    end if
    Establish S' as the solution for the time interval $[T_k, T_{k+1})$; i.e., for each $st_{ij}^* \in S'$ such that $T_k \leq st_{ij}^* < T_{k+1}$, set $st_{ij} = st_{ij}^*$ in the final schedule S;
    $k = k + 1$; $T_k = T_{k-1} + \Delta T$;
  end while
  return The schedule S;

5. GENETIC ALGORITHM

Algorithm 2 describes the GA proposed in this paper. The initial population is generated by a combination of random and heuristic chromosomes. Then, the algorithm iterates over a number of generations until the stopping criterion is satisfied. In each iteration, a new generation is built from the previous one by applying selection, crossover, mutation and replacement operators. The termination criterion is satisfied when the best chromosome of the population is not improved during a consecutive number of generations, or also if we find a solution with zero tardiness. In the following subsections we detail the different components of the GA.

5.1 Chromosome representation and evaluation

To codify chromosomes we have decided to use a permutation of the vehicles. We have also considered a matrix representation, but the experimental results were worse.

To create a schedule S from a permutation V, the idea is to sequentially schedule all vehicles of V, choosing for each vehicle the earliest starting time such that all constraints are met with respect to the previously scheduled vehicles. However, the constraint of maximum imbalance between any two lines may lead to some undesirable assignments of charging times. This is clarified with the following example.

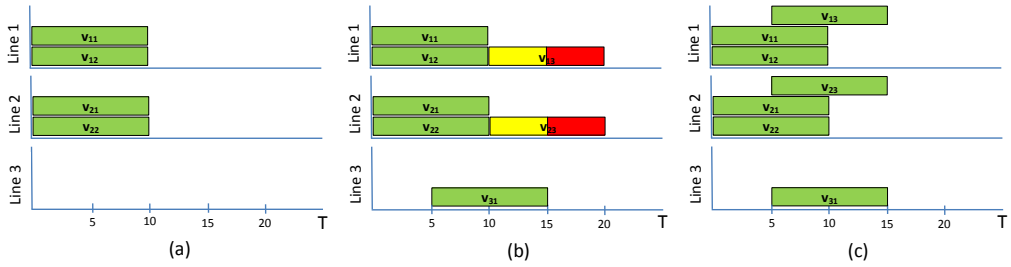


Figure 2: Example to illustrate the scheduler algorithm

Algorithm 2 The genetic algorithm

Require: A problem instance P
  Generate the initial population;
  while Not termination criterion is satisfied **do**
    Group all chromosomes randomly in pairs;
    Apply the crossover operator to each pair with probability *cr_prob*;
    Apply the mutation operator to each line of every offspring with probability *mut_prob*;
    Evaluate all new chromosomes;
    Choose chromosomes for the next generation with the replacement strategy;
  end while
  return The schedule S from the best chromosome;

Consider a problem with three vehicles (v_{11}, v_{12}, v_{13}) in L_1, three vehicles (v_{21}, v_{22}, v_{23}) in L_2 and one vehicle (v_{31}) in L_3. Arrival time is 0 for all vehicles except for v_{31}, being $t_{31} = 5$. Charging time is 10 for all vehicles. Due date is 10 for vehicles v_{11}, v_{12}, v_{21}, v_{22}, and 15 for vehicles v_{13}, v_{23}, v_{31}. The maximum number N of active charging points in any line is 3 and the maximum imbalance parameter Δ is 2/3 (therefore, at any given time point there should be at most a difference of two vehicles between any two lines).

Consider the permutation $(v_{11}, v_{12}, v_{21}, v_{22}, v_{13}, v_{23}, v_{31})$ and the partial schedule where the first four vehicles have already been scheduled (see Figure 2(a)). In this situation, when vehicles v_{13} and v_{23} are to be scheduled, both need to be scheduled at time 10 because if they are scheduled at time 5 then the maximum imbalance constraint with respect to line 3 is violated. Afterwards, v_{31} can be scheduled at time 5 and the result is the schedule of Figure 2(b) with tardiness 10 (we mark in red the portion of the charging time that surpasses the due date). However, notice that the scheduling of v_{31} breaks the previous "imbalance lock" in lines 1 and 2, and now both v_{13} and v_{23} could be reassigned to start charging at time 5, as indicated in Figure 2(c), to obtain a solution with no tardiness.

This example motivates us to design our scheduler algorithm so that vehicles can be reassigned in the following way. When we schedule the charging of a vehicle v' in line L' with charging time p' we choose the minimum starting time t' that fulfills all constraints. Then, we check if this new assignment breaks a previous imbalance lock, i.e. before the assignment of v' no vehicle could be scheduled in some other line because of the maximum imbalance with respect to line L', but after the assignment of v' it is possible to do it.

Algorithm 3 Scheduler algorithm

Require: A permutation of vehicles V
  $S \leftarrow \varnothing$;
  while S contains less vehicles than V **do**
    Let v' be the leftmost vehicle in V such that $v' \notin S$;
    Let t' be the minimum starting time for v' that fulfills all constraints, L' its line and p' its charging time;
    if the assignment of v' in the interval [t', t'+p'] breaks a previous imbalance lock **then**
      Unschedule (i.e. remove from S) all vehicles of lines different from L' such that its starting time is $\geq t'$;
    end if
    Schedule v' (i.e. append v' at the end of S) with starting time t';
  end while
  return The schedule S;

Formally, we can define that an assignment of a feasible starting time t' to a vehicle v' breaks a previous imbalance lock if and only if before the assignment it holds that $\exists t \in [t', t' + p')$ and $\exists i \in \{1, 2, 3\}, i \neq L'$ such that $\frac{N_i(t)+1-N_{L'}(t)}{N} > \Delta$, and after the assignment, with $N_{L'}$ updated in $[t', t' + p')$, it holds that $\frac{N_i(t)+1-N_{L'}(t)}{N} \leq \Delta$.

If this occurs, we propose to unschedule all vehicles of lines different from line L' such that its charging time starts at a time $\geq t'$, so that they may be rescheduled again with an earlier starting time. Therefore, as now the vehicles may be unscheduled, instead of sequentially scheduling all vehicles in V, we opted to schedule the leftmost vehicle from V such that its not yet scheduled (or has been unscheduled). Algorithm 3 shows the details of the procedure. Notice that the resulting schedule is always feasible.

Figure 3 shows a graphical representation of a schedule for an instance with 180 vehicles. We show the charging intervals for all vehicles assigned to each line, while in the bottom of the figure we report the maximum imbalance level among the lines. Cars that finished charging before their due date are colored in green, while cars that finished after their due date are colored in yellow and red (in red the portion after the due date, i.e. the tardiness of the vehicle).

5.2 Initial population

To create a diverse initial population we propose to use a combination of two dispatching rules together with some random chromosomes. One third of the population is created with each technique.

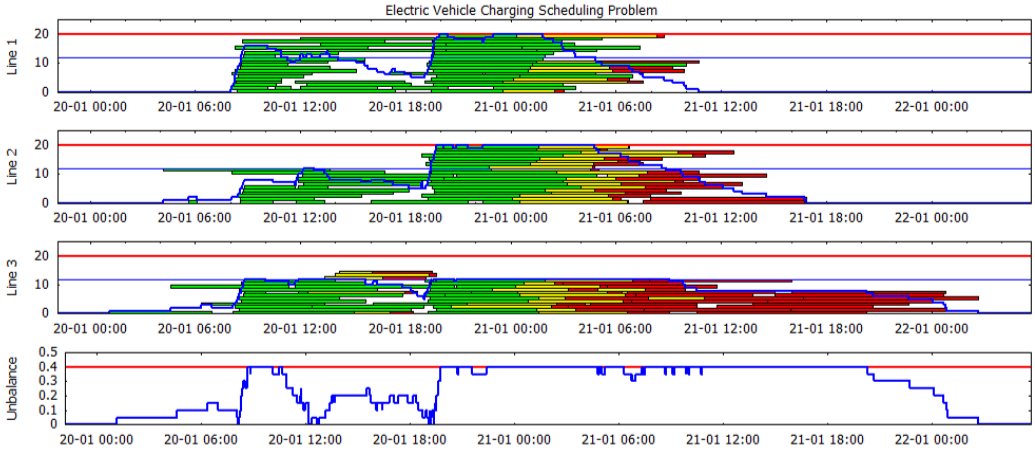


Figure 3: Example of a problem schedule. Green intervals represent charging times of vehicles that finish before their due date. Yellow and red intervals represent charging times of vehicles that finish after their due date (in yellow the portion before the due date and in red the portion after the due date, i.e. the tardiness of the vehicle).

The first dispatching rule is the Due Date rule, which sorts the vehicles in increasing order of its due date d_{ij}.

The second is the Apparently Tardiness rule, which is an adaptation of the rule used in [6]. This rule was used in many papers (see for example [9]) and is well-known for its effectiveness when minimizing tardiness. We propose to adapt this rule as follows: let $\Gamma(\alpha)$ be the earliest starting time for an unscheduled vehicle in the partial schedule α built so far. Then for all unscheduled vehicles that can start at $\Gamma(\alpha)$ a selection probability is calculated as

$$\Pi_j = \frac{1}{p_j} \exp\left[\frac{-\max(0, d_j - \Gamma(\alpha) - p_j}{g\overline{p}}\right] \qquad (8)$$

where $\overline{p}$ is the average charging time of vehicles and g is a look-ahead parameter to be fixed empirically, which is set here to 0.25 based on the experiments performed in [6]. The rule sorts all the vehicles in ascendent order of Π_j.

These two rules are deterministic but we need to create diverse chromosomes. Therefore, to select the next vehicle to be appended to the chromosome, firstly we sort the vehicles with the corresponding dispatching rule and then we perform a tournament selection. To this end, we randomly select a number $tSize$ of vehicles and we add to the chromosome the best of the chosen vehicles according to the ordering given by the dispatching rule. The tournament size $tSize$ is a relevant parameter, as if it is too large very similar chromosomes will be produced, whereas too small tournament sizes may result in almost random chromosomes.

5.3 Selection and replacement strategies

In the selection phase all chromosomes are grouped into pairs, and then the crossover operator is applied to each pair to produce two offspring solutions, which can then be modified by the mutation operator. Then, the replacement strategy is used to choose the chromosomes that will advance to the next generation. For each pair of parents and its two offspring, we choose the best two chromosomes such that they have a different tardiness. In this way we preserve the diversity avoiding to choose two identical chromosomes. Also notice that the best solution of a generation is always equal or better than the best solution of the previous generation. In case that the two parents and their two offspring have all the same tardiness, we choose two of them randomly.

5.4 Crossover operator

The crossover operator should generate chromosomes that inherit good characteristics of their parents and hopefully improve the value of the fitness function. In this paper we define three different crossover operators that combine two parents (P_1 and P_2) to produce two offspring (O_1 and O_2).

The first one is denoted LEX (Line Exchange Crossover), and it starts randomly selecting one line L_i. Then, it assigns to the offspring O_1 all vehicles of L_i in the same order as they appear in P_2, and the vehicles of the remaining two lines in the same order as in P_1. To create O_2 the vehicles of L_i are taken from P_1 and the remaining vehicles from P_2.

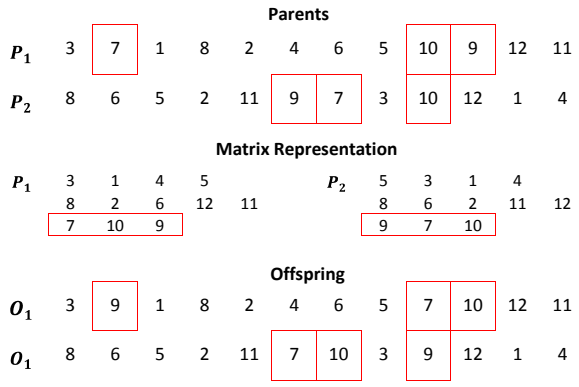


Figure 4: Example of LEX. Line 3 is randomly selected and its vehicles are exchanged in the offspring.

The second operator, denoted SBX (Starting-time Based Crossover) randomly selects a time t_0. Then, O_1 is created with all vehicles of P_1 such that they start charging before t_0. The chromosome is completed with the remaining vehicles in the same order as they appear in P_2. O_2 is created similarly.

Figures 4 and 5 show examples of operators *LEX* and *SBX* respectively. In these examples, the conversion from vector to matrix is performed with illustrative purposes only, as in practice it is not necessary.

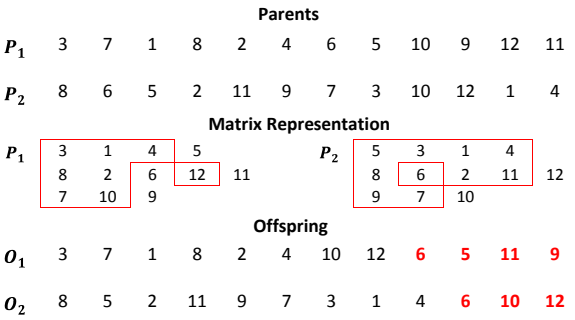


Figure 5: Example of *SBX*. Starting times of vehicles in P_1 are $\{0, 0, 0, 0, 0, 7, 10, 10, 0, 10, 7, 11\}$ and in P_2 are $\{0, 10, 0, 0, 7, 0, 0, 5, 9, 12, 0, 7\}$. Time $t_0 = 9$ is randomly selected, and then all vehicles that start charging before t_0 are taken from one parent and the remaining vehicles are taken from the other parent.

We also consider the standard one-point crossover operator (denoted *1PX*), which randomly chooses a position in the chromosome, and creates O_1 by choosing all vehicles of P_1 before that position, and completes O_1 by adding the remaining vehicles in the same relative order as in P_2. For creating O_2 the parents reverse their roles.

LEX and *SBX* are specifically designed to solve our particular problem and hence we expect that they are more efficient than the standard one-point crossover operator *1PX*.

Notice that no repairing mechanisms for infeasible solutions are needed, as the resulting chromosomes are evaluated (after the mutation step) by the scheduler algorithm detailed in Section 5.1, which always produces feasible schedules.

5.5 Mutation operator

The mutation operator should improve the diversity of the population and introduce new schemata into the chromosomes. We propose an operator that given a schedule, each line is mutated with some probability. If some line L_i is mutated, the operator selects a random subset of consecutive vehicles of that line, and randomly shuffles the order of these vehicles. Figure 6 shows an example of this operator in which one of the lines of the schedule is mutated.

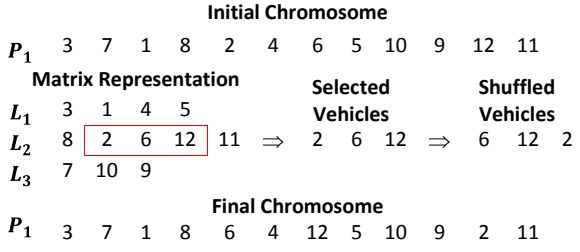


Figure 6: Example of the mutation operator. Each line is mutated with some probability (only L_2 in this example). A random subset of consecutive vehicles of the line is shuffled.

6. EXPERIMENTAL RESULTS

We have conducted an experimental study to evaluate our proposal and to compare it with the state-of-the-art in standard benchmarks. The GA has been implemented in C++ and the target machine is a PC with a Xeon E5520 processor and 24 GB RAM running Linux (SL 6.0).

6.1 Description of the benchmark set

We consider the benchmark set proposed in [6], where the charging station is installed in a car park with 180 spaces, with a time horizon of one day and different profiles of arrival times, demands and due dates based on the expected behavior of the users in some weekdays.

Two types of instances are defined. In type 1 instances 60 vehicles arrive at each line L_i along the day and demand charging, while in type 2 instances 108 vehicles arrive in L_1, 54 in L_2 and 18 in L_3, i.e., 60%, 30% and 10% respectively. So, in the later case we may expect that the scheduling algorithm has to control many situations of imbalance among the lines in order to build a feasible schedule. We also consider different values for the imbalance parameter Δ (0.2, 0.4, 0.6 and 0.8) and for the maximum number of vehicles that can be charging at the same time in a line N (20, 30 and 40). 30 instances were generated for each combination of type, Δ and N, so we have a total of 720 instances.

6.2 GA parameter settings

Firstly we have conducted a preliminary study to set some of the parameters of the GA. We have considered a set of 24 instances, in particular the first one of each combination of parameters. We have set the population size at 200 chromosomes and the stopping condition at 25 generations without improving the best solution found so far, or also if we find a solution with zero tardiness. These values are reasonable and by using them the computational time is adequate. Then, we tested different values for the remaining GA parameters, i.e. crossover probability, mutation probability for each line of the schedule, and tournament size for the dispatching rules used to create the initial population. Table 1 shows a summary of the tested values, indicating in bold the configuration that achieved the best average results.

Table 1: Values tested in the parameter tuning. Bold values indicate the best configuration found.

Parameter	Values tested
cr_prob	0.6, **0.8**, 1
mut_prob	0, **0.1**, 0.3
tSize	4, **8**, 12, 16

This configuration results in reasonable convergence patterns, as shown in Figure 7, which details the evolution of the best and average tardiness of the population for one run on an instance of *Type* 1, $N = 20$ and $\Delta = 0.2$.

6.3 Comparison of the crossover operators

In this section we compare the crossover operators *LEX*, *SBX* and *1PX* described in Section 5.4. Again, experiments were made in 24 instances: the first instance of each combination of parameters. GA is executed 30 times for each instance to obtain statistically significant results. Table 2 shows the results, in particular the average tardiness values grouped by instance type in both the static and dynamic

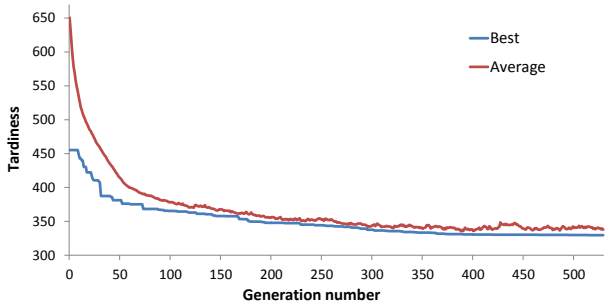


Figure 7: Evolution of the best and average tardiness of the population depending on the generation number, for one run on an instance of $Type$ 1, $N = 20$ and $\Delta = 0.2$.

problems. We notice that *SBX* operator always obtains the best results, while *LEX* operator seems the worst option.

To further compare the quality of the proposed crossover operators, we have done some statistical tests to analyze differences between them. Following [5], since we have multiple-problem analysis, we have used non-parametric statistical tests. First, we have run a Shapiro-Wilk test to confirm the non-normality of the data. Then we have used paired Wilcoxon signed rank tests to compare the average values in all the 48 instances considered (24 for the static problem and 24 for the dynamic one). In all these tests, the level of confidence used was 95% and the alternative hypothesis was "the difference between the errors of operator *SBX* and the other operator is smaller than 0". The p-values obtained with these tests (*LEX*: 1.061e-08, *1PX*: 0.0001584) show that there exist statistically significant differences between *SBX* and the other proposed operators in these instances.

We believe that *SBX* obtains better results than *1PX* because it is based on the phenotype instead of the genotype, and therefore it captures better the particular issues of our problem. On the other hand, *LEX* obtains worse results probably because the created offspring is too similar to their parents and hence the diversity of the population is lower.

6.4 Comparison with the state-of-the-art

In this section we compare the GA (using *SBX* operator) with the state-of-the-art method in this problem, which is the Problem Decomposition approach (PD) proposed in [6]. In this case we consider the full benchmark, i.e. the 720 instances in both the static and dynamic versions of the problem. Again, our GA is run 30 times for each instance and the best and average tardiness (in hours) are registered.

Table 3 shows the results of these experiments. We report, for each group of 30 instances, the sum of the tardiness of all 30 instances. It is remarkable that our GA obtains better average tardiness values than PD in all groups of instances in both the static and dynamic versions of the problem.

To further compare GA and PD, as similarly done in Section 6.3, we have run a Shapiro-Wilk test to confirm the non-normality of the data and then a paired Wilcoxon signed rank test to compare the average tardiness values obtained by GA and PD results in the 720 considered instances in the dynamic problem (which is arguably the most fair comparison). The p-value obtained from these tests (lower than 2.2e-16) show that there exist statistically significant differences between GA and PD, and then we can conclude that GA is significantly better than the state-of-the-art.

Table 2: Comparison between the proposed crossover operators in a set of 24 instances. We report the average tardiness in hours of 30 runs, grouped by instance type.

Ins.	Static			Dynamic		
	LEX	*SBX*	*1PX*	*LEX*	*SBX*	*1PX*
Type1	56.34	55.27	55.67	73.24	71.79	72.11
Type2	1157	1113	1118	1129	1119	1121

It is also essential to discuss computational times, as the algorithm is intended to be run in a real scenario. In the static problem the average time is 93 seconds per run, although the time varies depending on the instance parameters. In fact, instances with $\Delta = 0.2$ generally take more time because the imbalance constraints are much more restrictive and the problem is more difficult to solve. In the dynamic problem the average time is 325 seconds. However, this number accounts for the total running time along the 24-hour period. As the algorithm is run every two minutes, it is run 720 times in a 24-hour period, and therefore the average running time is only 0.45 seconds. Again, this number depends on the instance parameters, but in the worst case the average time is 2.75 seconds and so we can conclude that the algorithm is appropriate in a real-world setting.

In Table 3 we can also observe that the genetic algorithm reaches much better tardiness values for the static version than for the dynamic one. Although in absolute terms the reduction is similar in type 1 and type 2 instances, the improvements are specially remarkable in type 1 instances, as in relative terms the average reduction is nearly of 50%. We think that type 2 instances offer less room for improvement, as the difference of vehicle arrivals in each line (60% in L_1, 30% in L_2 and 10% in L_3) makes much more difficult to improve the schedule because of the imbalance constraints. Anyway, we can conclude that GA is able to exploit all the extra information about arrival times, charging times and due dates to reach better solutions.

7. CONCLUSIONS

In this work we addressed the problem of scheduling the charging of electric vehicles and we developed a genetic algorithm that obtains competitive results. The problem is defined in [6] and it is motivated by a real environment in which a number of vehicles may require charge from an electric system installed in a garage. The environment has a number of constraints, as a maximum number of vehicles that can be charged simultaneously and also imbalance constraints among the three lines of the three-phase electric feeder. We have considered two versions of the problem: the static version and the dynamic one. In the static version we know in advance the arrival time, charging time and due date of every vehicle, whereas in the dynamic version we do not have these data.

We have proposed a genetic algorithm with some operators specifically designed to efficiently deal with our problem. We have analyzed our proposal and compared it with the state-of-the-art algorithm for this problem, which is the Problem Decomposition approach proposed in [6], obtaining significantly better results. The scheduler algorithm, the crossover and mutation operators and the way of gen-

Table 3: Comparison of our GA with the state-of-the-art algorithm PD. Each value corresponds to the sum of the tardiness (in hours) of the 30 instances of each group.

Instance		PD	GA (Static)		GA (Dynamic)	
N	Δ		Best	Avg.	Best	Avg.
Type 1						
20	0.2	7720	5250	5442	6744	7142
	0.4	4240	2616	2680	3844	3977
	0.6	3849	2260	2300	3494	3569
	0.8	3807	2204	2239	3435	3502
30	0.2	1782	931	997	1262	1412
	0.4	490	80	92	343	375
	0.6	458	37	50	302	319
	0.8	458	35	49	300	317
40	0.2	646	337	364	464	511
	0.4	28	0	0	4	6
	0.6	9	0	0	3	3
	0.8	9	0	0	3	3
Avg.		1958	1146	1184	1683	1761
Type 2						
20	0.2	127614	123167	124380	123746	124599
	0.4	46254	44675	45263	45049	45461
	0.6	23008	20957	21206	21771	22075
	0.8	14808	12888	13031	14107	14337
30	0.2	72460	70304	71129	70825	71462
	0.4	21427	20378	20630	21043	21321
	0.6	8079	7093	7188	7837	8007
	0.8	4501	3551	3607	4325	4408
40	0.2	46096	44618	45216	45031	45455
	0.4	10932	9883	10011	10593	10799
	0.6	3520	2870	2917	3432	3518
	0.8	1659	888	923	1524	1569
Avg.		31696	30106	30458	30773	31084

erating a diverse initial population are contributions of this work, and we think that the good performance of the overall algorithm is due to them.

We have seen that the tardiness obtained while solving the static version of the problem is much lower than that of the dynamic version. This leads us to think that there may be better ways of dealing with the dynamic problem. Maybe if we could anticipate some car arrivals (for example having a probability distribution of the arrival of cars depending on the time of the day), then we may avoid some bottlenecks and hence schedule the cars in a more effective way. This may be the subject of further research.

We also plan to compare our results with those reported in [7]. Additionally, it would be interesting to design new and improved crossover operators for this problem, as well as compare them with other well-known operators from the literature, as for example PMX (Partially Matched Crossover) or CX (Cycle Crossover). Another research line could be designing local search methods to combine with the GA in order to improve its results.

Finally, we plan to consider additional characteristics of the real problem. For example, the users may pick up the vehicle before the declared due date, or the battery may get fully charged before the expected charging time p_{ij}. It would also be interesting to consider that the contracted power changes over time or that the vehicles can be charged at non constant rate.

8. ACKNOWLEDGMENTS

This research has been supported by the Spanish Government under research project TIN2013-46511-C2-2-P.

9. REFERENCES

[1] D. Dallinger. *Plug-in Electric Vehicles Integrating Fluctuating Renewable Electricity*. Kassel University Press, 2013.

[2] EDSO. Position paper on electric vehicles charging infrastructure. Technical report, European Distribution system Operators for Smart Grids (EDSO), 10 April 2012.

[3] L. Gan, U. Topcu, and S. Low. Optimal decentralized protocol for electric vehicle charging. *IEEE Transaction on Power Systems*, 28(2):940–951, 2013.

[4] J. García-Villalobos, I. Zamora, J. San Martín, F. Asensio, and V. Aperribay. Plug-in electric vehicles in electric distribution networks: A review of smart charging approaches. *Renewable and Sustainable Energy Reviews*, 38:717–731, 2014.

[5] S. García, A. Fernández, J. Luengo, and F. Herrera. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences*, 180:2044–2064, 2010.

[6] A. Hernandez-Arauzo, J. Puente, M. A. Gonzalez, R. Varela, and J. Sedano. Dynamic scheduling of electric vehicle charging under limited power and phase balance constraints. In *Proceedings of the 7th Scheduling and Planning Applications Workshop (SPARK 2013)*, pages 1–8, 2013.

[7] A. Hernandez-Arauzo, J. Puente, R. Varela, and J. Sedano. Electric vehicle charging under power and balance constraints as dynamic scheduling. *Computers & Industrial Engineering*, Accepted Manuscript (unedited version) available online: 13-APR-2015. DOI: 10.1016/j.cie.2015.04.002, 2015.

[8] J. Kang, S. J. Duncan, and D. N. Mavris. Real-time scheduling techniques for electric vehicle charging in support of frequency regulation. *Procedia Computer Science*, 16:767–775, 2013.

[9] S. Kaplan and G. Rabadi. Exact and heuristic algorithms for the aerial refueling parallel machine scheduling problem with due date-to-deadline window and ready times. *Computers & Industrial Engineering*, 62(1):276–285, 2012.

[10] Z. Ma, D. Callaway, and I. Hiskens. Decentralized charging control of large populations of plug-in electric vehicles. *IEEE Transactions on Control Systems Technology*, 21(1):67–78, 2013.

[11] J. Sedano, M. Portal, A. Hernandez Arauzo, J. Villar, J. Puente, and R. Varela. Sistema de control autónomo para distribución de energía en una estación de carga de vehículos eléctricos: diseño y operación. Technical report, Instituto Tecnológico de Castilla y León ITCL, 2012.

[12] J. Yang, L. He, and S. Fu. An improved pso-based charging strategy of electric vehicles in electrical distribution grid. *Applied Energy*, 128:82–92, 2014.