



UNIVERSIDAD SIMÓN BOLÍVAR
DECANATO DE ESTUDIOS DE POSTGRADO
COORDINACIÓN DE POSTGRADO EN CIENCIAS DE LA COMPUTACIÓN
MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN

TRABAJO DE GRADO

**CONSULTAS SKYLINE ESPACIALES SOBRE DATOS
DINÁMICOS**

por

Fabiola Regina Di Bartolo Lara

Febrero 2013



UNIVERSIDAD SIMÓN BOLÍVAR
DECANATO DE ESTUDIOS DE POSTGRADO
COORDINACIÓN DE POSTGRADO EN CIENCIAS DE LA COMPUTACIÓN
MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN

**CONSULTAS SKYLINE ESPACIALES SOBRE DATOS
DINÁMICOS**

Trabajo de Grado presentado a la Universidad Simón Bolívar por
Fabiola Regina Di Bartolo Lara
como requisito parcial para optar al grado académico de
Magíster en Ciencias de la Computación
Con la asesoría del prof.
Marlene Goncalves

Febrero 2013



UNIVERSIDAD SIMÓN BOLÍVAR
DECANATO DE ESTUDIOS DE POSTGRADO
COORDINACIÓN DE POSTGRADO EN CIENCIAS DE LA COMPUTACIÓN
MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN

CONSULTAS SKYLINE ESPACIALES SOBRE DATOS DINÁMICOS

Por: Di Bartolo Lara Fabiola Regina
Carnet No.: 09-87324

Este Trabajo de Grado ha sido aprobado en nombre de la Universidad Simón Bolívar por el siguiente jurado examinador:

Soraya Abad Pota

Presidente(a)
Prof. Soraya Abad

Miembro Externo

Prof. Elsa Tovar
Universidad de Carabobo

Marlene Goncalves

Miembro Principal – Tutor
Prof. Marlene Goncalves

Sartenejas, 22 de abril de 2013

DEDICATORIA

Con mucho amor a todos mis seres queridos por apoyarme y creer en mí siempre. Querer y creer es poder, así que la mente funciona mejor sin límites!

AGRADECIMIENTOS

A Dios por guiarme, ayudarme a estar en pie ante los tropiezos y acompañarme siempre.

A mis padres Solange Lara y Gaetano Di Bartolo quienes me brindaron las bases que me hacen ser quién soy hoy en día, por su amor, por creer en mi ciegamente, por sus consejos y apoyarme incondicionalmente día tras día.

A Oscar Flores, compañero de esta travesía, por haber entendido que las noches, madrugadas y fines de semana eran para estudiar y aún así regalarme tranquilidad en momentos difíciles, por transmitirme su amor con miradas y sonrisas, por formar un hogar y a la vez, ser mi escape y complemento.

A mi familia por iluminar mis pasos y darle a mi vida música y color. A mis amigos por compartir con mucha alegría y sin reproches los pequeños ratos libres.

A mi tutora Marlene Goncalves quién con motivación y sabiduría, supo guiarme por el camino correcto, demostrando que con perseverancia una pequeña idea se convierte en un trabajo de grado.

A mis compañeros de maestría y amigos, por hacer divertidas las horas de estudio. A mis profesores, por enseñarme a estar en continuo aprendizaje.

A mis compañeros de Gram & Asociados, WTFE y Zapacos por brindarme una segunda casa. A mis jefes por hacer posible cada etapa de este sueño.

Gracias a todos!



UNIVERSIDAD SIMÓN BOLÍVAR
 DECANATO DE ESTUDIOS DE POSTGRADO
 COORDINACIÓN DE POSTGRADO EN CIENCIAS DE LA COMPUTACIÓN
 MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN
**CONSULTAS SKYLINE ESPACIALES SOBRE DATOS
DINÁMICOS**

Por: Di Bartolo Lara Fabiola Regina

Carnet No.: 09-87324

Tutor: Goncalves Marlene

Febrero 2013

RESUMEN

Las Consultas *Skyline* permiten expresar preferencias de minimización o maximización de los atributos sobre un conjunto de datos. Estas preferencias inducen un orden parcial sobre los datos, así que el problema de conseguir el *skyline* es análogo a obtener los elementos *maximales* de un CPO. Las Consultas *Skyline* son el centro de numerosos trabajos de investigación, no sólo por su utilidad en problemas de toma de decisiones, sino por su compleja evaluación. Recientemente, surgieron las consultas SSQ que emplean el *skyline* en grandes conjuntos de datos espaciales, éstas permiten expresar preferencias de minimización en las distancias de los objetos a ciertos puntos de referencia. No obstante, las SSQ no combinan esas preferencias con preferencias sobre datos no espaciales y altamente cambiantes. En este trabajo de grado, fueron extendidas las consultas SSQ a las consultas DSSQ, que resuelven este problema. En este sentido, se propusieron cuatro algoritmos que evalúan las consultas DSSQ permitiendo cambios de posición de algún punto de referencia y la modificación de un atributo altamente cambiante. Los algoritmos 2B2S e IB2S fueron diseñados como soluciones básicas, diferenciándose en que 2B2S procesa cada vez el *skyline* y IB2S lo actualiza al existir cambios en los objetos. El algoritmo VC2S+ es una adaptación del VCS² [1] que actualiza el *skyline* en presencia de movimiento de un punto de referencia. El algoritmo CD2S es una alternativa para el procesamiento continuo de estas consultas ya que el *skyline* es actualizado en ambos casos. En los resultados de los experimentos realizados, se observó que en la mayoría de los casos, las menores medias en tiempo de ejecución corresponden al algoritmo CD2S, siendo en el mejor de los casos hasta un cuarto y tercio del tiempo de estos algoritmos, a la vez que las comparaciones se reducen hasta más de un orden de magnitud.

Palabras claves: consultas de preferencias, *skyline* espacial, objetos móviles, atributos altamente cambiantes, consultas continuas.

ÍNDICE GENERAL

DEDICATORIA	iv
AGRADECIMIENTOS	v
RESUMEN	vii
ÍNDICE GENERAL	vii
ÍNDICE DE TABLAS	xii
ÍNDICE DE FIGURAS	xiv
GLOSARIO	xvii
INTRODUCCIÓN	1
CAPÍTULO I. PLANTEAMIENTO DEL PROBLEMA	6
1.1. Motivación	7
1.2. Definiciones preliminares	16
1.3. Definición del problema	21
CAPÍTULO II. EL <i>SKYLINE</i> , TRABAJOS RELACIONADOS	24
2.1. Consultas <i>skyline</i>	27
2.2. Consultas <i>skyline</i> con datos espaciales	29
2.3. Otras consultas espaciales	33
2.4. Consultas <i>skyline</i> en ambientes cambiantes	34
2.5. Consultas <i>skyline</i> no tradicionales en ambientes cambiantes	37
CAPÍTULO III. SOLUCIONES AL PROBLEMA DE DSS	39
3.1. Representación del problema	39
3.1.1. Simulador	41
3.1.2. Ejemplo de posibles simulaciones	42
3.1.2.1. Ejemplo I: Dimensiones sólo espaciales	42

3.1.2.2. Ejemplo II: Dimensiones espaciales y no espaciales	44
3.1.3. Preliminares	45
3.1.3.1. BNL	45
3.1.3.2. SFS	46
3.1.3.3. VS^2 y VCS^2	46
3.2. Soluciones basadas en BNL	49
3.2.1. Ejemplo de ejecución del algoritmo 2B2S	49
3.2.1.1. Ejemplo I: Dimensiones sólo espaciales	49
3.2.1.2. Ejemplo II: Dimensiones espaciales y no espaciales	50
3.2.2. Estructuras utilizadas por el algoritmo 2B2S	50
3.2.3. Descripción del algoritmo 2B2S	51
3.2.3.1. Modo inicial	51
3.2.3.2. Modo actualización de estatus	53
3.2.3.3. Modo actualización de desplazamiento	53
3.2.4. Algoritmo IB2S	54
3.2.4.1. Modo actualización de estatus	54
3.3. Algoritmo VC2S+	56
3.3.1. Ejemplo de ejecución del algoritmo VC2S+	57
3.3.1.1. Ejemplo I: Dimensiones sólo espaciales	57
3.3.1.2. Ejemplo II: Dimensiones espaciales y no espaciales	59
3.3.2. Estructuras utilizadas por el algoritmo VC2S+	60
3.3.3. Descripción del algoritmo VC2S+	61
3.3.3.1. Modo inicial	61
3.3.3.2. Modo actualización de estatus	65
3.3.3.3. Modo actualización de desplazamiento	65
3.4. Solución propuesta	68
3.4.1. Ejemplo de ejecución del algoritmo CD2S	68
3.4.1.1. Ejemplo I: Dimensiones sólo espaciales	68
3.4.1.2. Ejemplo II: Dimensiones espaciales y no espaciales	70
3.4.2. Estructuras utilizadas por el algoritmo CD2S	71
3.4.3. Descripción del algoritmo CD2S	72
3.4.3.1. Modo inicial	72
3.4.3.2. Modo actualización de estatus	75
3.4.3.3. Modo actualización de desplazamiento	79

CAPÍTULO IV.DISEÑO DEL ESTUDIO EXPERIMENTAL	82
4.1. Configuraciones	84
4.2. Clasificación de los casos de prueba	85
4.3. Simulaciones	85
4.4. Arquitectura utilizada	86
4.5. Métricas, indicadores y medidas	87
CAPÍTULO V. ESTUDIO EXPERIMENTAL	90
5.1. Experimento I	90
5.1.1. Metodología	91
5.1.2. Resultados	91
5.1.2.1. Estudio del tiempo de los algoritmos	92
5.1.2.2. Estudio de la cardinalidad del <i>skyline</i>	94
5.2. Experimento II	96
5.2.1. Metodología	96
5.2.2. Resultados	96
5.2.2.1. Estudio del tiempo de los algoritmos	97
5.2.2.2. Estudio de la cardinalidad del <i>skyline</i>	98
5.3. Experimento III	101
5.3.1. Metodología	101
5.3.2. Resultados	102
5.3.2.1. Estudio del tiempo de los algoritmos	102
5.3.2.2. Estudio de los patrones de desplazamiento y la actividad en el <i>skyline</i>	104
5.4. Experimento IV	107
5.4.1. Metodología	107
5.4.2. Resultados	108
5.4.2.1. Estudio del tiempo de los algoritmos	108
5.4.2.2. Estudio del porcentaje de cambio y la actividad en el <i>skyline</i>	110
5.5. Experimento V	112
5.5.1. Metodología	112
5.5.2. Resultados	112
5.5.2.1. Estudio del tiempo de los algoritmos	113
5.5.2.2. Estudio del área del <i>convex hull</i>	114

5.6. Experimento VI	116
5.6.1. Metodología	116
5.6.2. Resultados	117
5.6.2.1. Estudio del tiempo de los algoritmos	117
5.6.2.2. Estudio de las comparaciones y cardinalidad del <i>skyline</i> .	119
5.6.2.3. Estudio de los candidatos al <i>skyline</i>	121
CAPÍTULO VI.CONCLUSIONES Y RECOMENDACIONES	123
REFERENCIAS	131
APÉNDICE A. ARCHIVOS DE ENTRADA	139
APÉNDICE B. SIMULADOR	142
APÉNDICE C. CREACIÓN DE LA REGIÓN DE BÚSQUEDA	144
APÉNDICE D. CARACTERÍSTICAS DE LOS ALGORITMOS	146
APÉNDICE E. TABLAS DE RESULTADOS	150
APÉNDICE F. SIMULACIONES	171
APÉNDICE G. PROPIEDADES ESPACIALES	173
APÉNDICE H. RESULTADOS COMPLEMENTARIOS	176
H.1. Experimento I	176
H.1.1. Tiempo detallado	176
H.1.2. Cardinalidad del <i>skyline</i> no espacial	178
H.1.3. Menores y mayores tiempos de ejecución	178
H.1.4. Cambios en el <i>skyline</i>	179
H.2. Experimento II	181
H.2.1. Menores y mayores tiempos de ejecución	181
H.2.2. Histograma	184
H.3. Experimento III	186
H.3.1. Menores y mayores tiempos de ejecución	186
H.3.2. Candidatos en el <i>skyline</i>	189
H.3.3. Comparaciones de dominancia	190

H.4.	Experimento IV	193
H.4.1.	Tiempo y cardinalidad del <i>skyline</i>	193
H.4.2.	Menores y mayores tiempos de ejecución	196
H.4.3.	Candidatos en el <i>skyline</i>	199
H.4.4.	Comparaciones de dominancia	199
H.5.	Experimento V	201
H.5.1.	Menores y mayores tiempos de ejecución	201
H.5.2.	Histograma	201
H.5.3.	Comparaciones de dominancia y cambios en el <i>skyline</i>	204
H.6.	Experimento VI	205
H.6.1.	Menores y mayores tiempos de ejecución	205
H.6.2.	Objetos comparados	205
H.6.3.	Estudio de memoria	207
H.6.4.	Actividad en el <i>skyline</i>	209

ÍNDICE DE TABLAS

1.1. Características de los datos	9
1.2. Características de los datos	11
2.1. Complejidad en tiempo de algunos algoritmos para el <i>skyline</i> [2]	25
4.1. Casos generados para la inicialización de los algoritmos	84
4.2. Clasificación de los casos de prueba	85
5.1. Resultados por lotes de 50 simulaciones	91
5.2. Resultados por lotes de 50 simulaciones	97
5.3. Resultados por lotes de 50 simulaciones	102
5.4. Resultados por lotes de 50 simulaciones	108
5.5. Resultados por lotes de 50 simulaciones	113
5.6. Resultados por lotes de 50 simulaciones	117
D.1. Cuadro comparativo de los algoritmos (1 de 4)	146
D.2. Cuadro comparativo de los algoritmos (2 de 4)	147
D.3. Cuadro comparativo de los algoritmos (3 de 4)	148
D.4. Cuadro comparativo de los algoritmos (4 de 4)	149
E.1. Resultados por lotes para la configuración 1 (combinado-aleatorio-10) . . .	151
E.2. Resultados por lotes para la configuración 1 (combinado-aleatorio-20) . . .	152
E.3. Resultados por lotes para la configuración 2 (combinado-aleatorio-10) . . .	153
E.4. Resultados por lotes para la configuración 2 (combinado-aleatorio-20) . . .	154
E.5. Resultados por lotes para la configuración 2 (movimiento-N/A-5)	155
E.6. Resultados por lotes para la configuración 2 (movimiento-N/A-10)	156
E.7. Resultados por lotes para la configuración 2 (movimiento-N/A-15)	157
E.8. Resultados por lotes para la configuración 2 (movimiento-N/A-20)	158
E.9. Resultados por lotes para la configuración 2 (movimiento-N/A-25)	159
E.10. Resultados por lotes para la configuración 2 (cambioEstatus-10 %-N/A) . .	160
E.11. Resultados por lotes para la configuración 2 (cambioEstatus-30 %-N/A) . .	161
E.12. Resultados por lotes para la configuración 2 (cambioEstatus-aleatorio-N/A)	162
E.13. Resultados por lotes para la configuración 2 (cambioEstatus-70 %-N/A) . .	163
E.14. Resultados por lotes para la configuración 2 (cambioEstatus-90 %-N/A) . .	164

E.15. Resultados por lotes para la configuración 3 (combinado-aleatorio-10)	165
E.16. Resultados por lotes para la configuración 3 (combinado-aleatorio-20)	166
E.17. Resultados por lotes para la configuración 4 (combinado-aleatorio-10)	167
E.18. Resultados por lotes para la configuración 5 (combinado-aleatorio-10)	168
E.19. Resultados por lotes para la configuración 6 (combinado-aleatorio-10)	169
E.20. Resultados por lotes para la configuración 7 (combinado-aleatorio-50)	170
F.1. Ejecuciones realizadas	171
H.1. Estadísticas de los histogramas	186
H.2. Estadísticas de los histogramas	204

ÍNDICE DE FIGURAS

1.1.	Ambiente inicial	9
1.2.	Ambiente transformado	11
3.1.	Ejemplo I, <i>modo inicial</i> (simulación 0)	43
3.2.	Ejemplo I, <i>modo actualización de estatus</i> (simulación 1)	43
3.3.	Ejemplo I, <i>modo actualización de desplazamiento</i> (simulación 2)	43
3.4.	Ejemplo II, <i>modo inicial</i> (simulación 0)	44
3.5.	Ejemplo II, <i>modo actualización de estatus</i> (simulación 1)	44
3.6.	Ejemplo II, <i>modo actualización de desplazamiento</i> (simulación 2)	45
3.7.	Patrones de cambio del <i>convex hull</i> de Q cuando q cambia a q' . Fuente [1] .	47
3.8.	Ejemplo I, <i>modo inicial</i> (simulación 0)	58
3.9.	Ejemplo I, <i>modo actualización de estatus</i> (simulación 1)	58
3.10.	Ejemplo I, <i>modo actualización de desplazamiento</i> (simulación 2)	59
3.11.	Ejemplo II, <i>modo inicial</i> (simulación 0)	59
3.12.	Ejemplo II, <i>modo actualización de estatus</i> (simulación 1)	59
3.13.	Ejemplo II, <i>modo actualización de desplazamiento</i> (simulación 2)	60
3.14.	Ejemplo I, <i>modo inicial</i> (simulación 0)	69
3.15.	Ejemplo I, <i>modo actualización de estatus</i> (simulación 1)	69
3.16.	Ejemplo I, <i>modo actualización de desplazamiento</i> (simulación 2)	70
3.17.	Ejemplo II, <i>modo inicial</i> (simulación 0)	70
3.18.	Ejemplo II, <i>modo actualización de estatus</i> (simulación 1)	71
3.19.	Ejemplo II, <i>modo actualización de desplazamiento</i> (simulación 2)	71
5.1.	Tiempo de ejecución promedio por lote	92
5.2.	Comparaciones efectuadas en promedio por lote	93
5.3.	Composición promedio del <i>skyline</i> por lote	95
5.4.	Tiempo promedio de ejecución por lote	97
5.5.	Comparaciones efectuadas en promedio por lote	99
5.6.	Composición promedio del <i>skyline</i> por lote	100
5.7.	Tiempo de ejecución promedio por lote	103
5.8.	Casos de desplazamiento en promedio por lote	105

5.9. Cambios en el <i>skyline</i> en promedio por lote (Escala logarítmica)	106
5.10. Tiempo promedio de ejecución por lote	109
5.11. Cambios en el <i>skyline</i> en promedio por lote (Escala logarítmica)	111
5.12. Resultados generales	113
5.13. Composición promedio del <i>skyline</i> por lote	115
5.14. Tiempo promedio por lote y modo de ejecución	118
5.15. Comparaciones efectuadas en promedio	119
5.16. Composición promedio del <i>skyline</i> por lote	120
5.17. Candidatos al <i>skyline</i> en promedio por lote	122
A.1. Ejemplo del archivo aConf	139
A.2. Ejemplo del archivo aPtoRef	140
A.3. Ejemplo del archivo aDimNoEsp	140
C.1. Ejemplo de regiones de búsqueda	145
G.1. Propiedades espaciales	173
H.1. Tiempo promedio por lote y modo de ejecución	177
H.2. Porcentaje de lotes en que un algoritmo es el mejor, vecesTMin	178
H.3. Porcentaje de lotes en que un algoritmo es el peor, vecesTMax	179
H.4. Cambios realizados en el <i>skyline</i> en promedio por lote (Escala logarítmica)	180
H.5. Porcentaje de lotes en que un algoritmo es el mejor, vecesTMin	181
H.6. Porcentaje de lotes en que un algoritmo es el peor, vecesTMax	182
H.7. Distribución del tiempo promedio de ejecución por lote para VC2S+	183
H.8. Histogramas del tiempo de ejecución (Muestra=480 lotes, Intervalos=10) .	185
H.9. Porcentaje de lotes en que un algoritmo es el mejor, vecesTMin	187
H.10. Porcentaje de lotes en que un algoritmo es el peor, vecesTMax	188
H.11. Candidatos al <i>skyline</i> en promedio de todas las variantes por lote	189
H.12. Comparaciones efectuadas en promedio por lote	191
H.13. Composición promedio del <i>skyline</i> por lote	192
H.14. Composición promedio del <i>skyline</i> por lote	194
H.15. Distribución del tiempo promedio de ejecución por lote para VC2S+	195
H.16. Porcentaje de lotes en que un algoritmo es el mejor, vecesTMin	197
H.17. Porcentaje de lotes en que un algoritmo es el peor, vecesTMax	198
H.18. Candidatos al <i>skyline</i> en promedio de todas las variantes por lote	199
H.19. Comparaciones efectuadas en promedio por lote	200

H.20.Porcentaje de lotes en que un algoritmo es el mejor <code>vecesTMin</code> o el peor <code>vecesTMax</code>	202
H.21.Histogramas del tiempo de ejecución (Muestra=240 lotes, Intervalos=10) .	203
H.22.Resultados generales	204
H.23.Porcentaje de lotes en que un algoritmo es el mejor, <code>vecesTMin</code>	205
H.24.Porcentaje de lotes en que un algoritmo es el peor, <code>vecesTMax</code>	206
H.25.Objetos comparados en promedio	206
H.26.Memoria promedio por lote	207
H.27.Cambios en el <i>skyline</i> en promedio por lote	209

GLOSSARIO

2B2S	Algoritmo básico basado en BNL (<i>Basic BNL for DSS</i>).
Algoritmo progresivo	(<i>Skyline</i>). Algoritmos que producen resultados a lo largo de su ejecución ejecutado, es decir, resultados iniciales antes de terminar de evaluar el conjunto <i>skyline</i> completo
Ambiente	En las consultas DSSQ, es una instancia de todos los parámetros.
Atributo altamente cambiante	Característica no estático de un objeto que cambia de valor en el tiempo.
Atributo no espacial	Característica de un objeto que no esta relacionada con datos espaciales. Comúnmente son estáticos.
BNL	Algoritmo <i>Block Nested loop</i> .
CD2S	Algoritmo de evaluación progresiva y continua basado en el <i>convex hull</i> de los puntos de referencia y en los patrones de cambio que utiliza <i>VCS</i> ² (<i>Continuous Dynamic Spatial Skyline</i>).
Caso de prueba	En los experimentos, es el ambiente inicial generado aleatoriamente.
Combinación	(Experimentos). Corresponde a los valores: cantidad de puntos de referencia y número de dimensiones no espaciales empleados para una consulta.
Comparación de dominancia	Comparación realizada entre las dimensiones de dos objetos para determinar la dominancia.
Configuración	(Experimentos). Instancia de los parámetros empleados para la generación de un ambiente inicial.
Conjunto Skyline	Es el conjunto de tuplas que satisfacen la ejecución de un operador lógico <i>skyline</i> . Matemáticamente, son el conjunto de maximales del orden parcial establecido por las dimensiones de un <i>skyline</i> .
Convex Hull	Envolvente convexa. Es el polígono convexo formado por un conjunto de puntos dado en el plano euclíadiano, cuyos vértices son algunos de estos puntos, los restantes estarán dentro del polígono.
CPO	Conjunto parcialmente ordenado, es la formalización de un orden de los elementos en un conjunto, para lo cual se tiene una relación binaria de orden parcial
Diagrama de Voronoi	Construcción geométrica que divide el espacio en diferentes regiones del plano Euclíadiano estudiada por el matemático Georgy Voronoi. Cada punto tiene una region correspondiente denominada celda de Voronoi, que consiste en todos los puntos cercanos a ese punto que a cualquier otro.
Dimensión espacial	Corresponde a una preferencia espacial evaluada para un punto de referencia. Sus valores son las distancias de cada objeto al punto de referencia.
Dimensión no espacial	Corresponde a una preferencia realizada sobre un atributo no espacial.
Distancia Euclíadiana	Métrica de distancia más conocida. Entre dos puntos $x = (x_1, x_2)$ y $y = (y_1, y_2)$ es igual a $d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$
Dominancia	(<i>Skyline</i>). Un objeto a domina a b si $a \succ b$ en el orden parcial establecido por las dimensiones de un <i>skyline</i> .

DSS	Problema del <i>Skyline Espacial</i> sobre Datos Dinámicos (<i>Dynamic Spatial Skyline</i>).
DSSQ	Consulta <i>Skyline Espacial</i> sobre Datos Dinámicos (<i>Dynamic Spatial Skyline Query</i>).
Espacio de búsqueda	Conjunto de soluciones que existen para un mismo problema.
Grafo Delaunay	Triangulación de Delaunay la cual es el problema dual al diagrama de Voronoi de un conjunto de puntos.
IB2S	Algoritmo mejorado basado en BNL. <i>Improved BNL for DSS</i> .
Maximal	En un conjunto parcialmente ordenado CPO, un maximal es un elemento para el cual no existe otro elemento mayor que lo siga en el orden parcial.
Modo de ejecución	Forma escogida para la ejecución de los algoritmos propuestos. Depende de la transformación del ambiente ocurrida.
Modo actualización de estatus	Modo de ejecución empleado para la evaluación de la consulta tras una transformación del ambiente por cambios en los objetos.
Modo actualización de desplazamiento	Modo de ejecución empleado para la evaluación de la consulta tras una transformación del ambiente por desplazamiento del punto de referencia móvil.
Modo inicial	Modo de ejecución empleado para la evaluación de la consulta inicial.
Lote	(Experimentos). Conjunto de 50 simulaciones consecutivas. Partiendo de un caso de prueba, se ejecuta una consulta inicial y 49 consultas a raíz de transformaciones generadas en el ambiente.
Objeto	Instancia de los datos. Conjunto de valores que forman la unidad mínima de información.
Objeto espacial	Objeto que tiene coordenadas espaciales.
Optimalidad de Pareto	Problema de optimización multi-objetivo definido por el economista Vilfredo Pareto.
Punto de referencia	Coordenadas ingresadas a la consulta para indicar la localización de un punto de interés (<i>Query point</i>). A partir del punto de referencia se establece una preferencia espacial de minimización de la distancia de los objetos a este punto.
Punto de referencia móvil	Punto de referencia no estático que se encuentra en continuo desplazamiento.
SFS	Algoritmo <i>Sort Filter Skyline</i>
Variante	(Experimentos). Variante es un conjunto de simulaciones realizadas, que comparten los mismos parámetros, para una configuración en particular.
VCS²	Algoritmo de evaluación continua basado en Voronoi (<i>Voronoi-Based Continuous SSQ</i>).
VC2S+	Adaptación de los algoritmos <i>VS²</i> y <i>VCS²</i> al problema DSS (<i>Voronoi-Based Continuous SSQ Plus</i>).
VS²	Algoritmo basado en Voronoi (<i>Voronoi-Based Spatial Algorithm</i>).
Simlacióñ	(Experimentos). Alteración o transformación ficticia del ambiente. Produce una nueva consulta a ser evaluada.
SSQ	Consultas <i>Skyline Espaciales</i> (<i>Spatial Skyline Query</i>).
Transformación del ambiente	Operación que genera un nuevo ambiente con respecto a la instancia anterior. En el problema DSS, puede generarse por el desplazamiento del punto de referencia móvil o por cambios en los objetos (nuevos valores en el atributo altamente cambiante).

INTRODUCCIÓN

Cada día aumenta la necesidad de realizar consultas con preferencias particulares que soporten y faciliten la toma de decisiones, ofreciendo un rango reducido de opciones que satisfagan esas preferencias. En este sentido, han surgido las consultas *skyline* [3], cuyo objetivo es obtener la mejor aproximación a los criterios de preferencia que el usuario ha provisto.

El *skyline* se basa en la optimalidad de Pareto, en la maximización de funciones multicriterio o el problema del vector máximo [2, 4, 5] para establecer un orden parcial entre las objetos del conjunto de datos de entrada. Este orden es inducido por las preferencias dadas por el usuario que permiten la maximización o minimización simultánea de los atributos de dicho conjunto de datos. Los objetos que forman parte del *skyline* son los pertenecientes al conjunto *maximal*¹ de ese orden parcial.

El conjunto *skyline* generalmente tiene una cardinalidad mucho menor que el conjunto de datos sobre el cual se calculó [6]. Sin embargo, el *skyline* consume altos recursos de procesamiento debido a las comparaciones necesarias que deben ser realizadas entre los objetos para obtener el conjunto, por lo que es importante la existencia de algoritmos de evaluación que reduzcan estas comparaciones.

Recientemente se ha extendido la aplicación del problema *skyline* a conjuntos con una gran cantidad de datos espaciales. Una estrategia para filtrar esta información bajo algún criterio de preferencia, es mediante las consultas *skyline* espaciales (*spatial skyline queries*, SSQ), las cuales permiten definir como preferencias la minimización de las distancias

¹En un conjunto parcialmente ordenado CPO, un maximal es un elemento para el cual no existe otro elemento mayor que lo siga en el orden parcial.

de estos objetos a ciertos puntos de referencia deseados [7]. El procesamiento de estas consultas es más complejo que el *skyline* tradicional ya que cada comparación entre los objetos requiere el cálculo de estas distancias a los puntos de referencia. Estas distancias pueden calcularse de forma dinámica, por lo que estos puntos de referencia pueden variar de posición.

Los consultas *skyline* espaciales pueden ser adaptadas para considerar datos no espaciales y datos que cambien continuamente en el tiempo, en función de algún atributo dinámico de los mismos. Esta extensión es propuesta en este trabajo de grado, denominada consultas *skyline* espaciales sobre datos dinámicos (*dynamic spatial skyline queries*, DSSQ), las cuales tienen múltiples aplicaciones. En tareas militares, pueden ser utilizadas para determinar las estaciones enemigas cercanas a los pelotones que aún no han sido controladas y que son más propensas a un ataque, a partir de las coordenadas de los pelotones de soldados en los campos enemigos. En situaciones de emergencia o desastres naturales, para encontrar las zonas cercanas altamente vulnerables que no han sido atendidas, tomando las zonas registradas como peligrosas y la ubicación terrestre, aérea o marítima de las brigadas de rescate.

Las DSSQ también pueden ser empleadas en sistemas de reservaciones que a la vez realicen recomendaciones a los usuarios. Las recomendaciones se construyen en base a las opciones que cumplan con algún propósito indicado por el usuario. Estas corresponden a las mejores opciones posibles según las preferencias definidas por el usuario y su ubicación actual.

Un usuario a través de su dispositivo móvil que identifique su posición, puede estar interesado en realizar una reservación con algún propósito, bien sea almorzar, ver una película u hospedarse, para lo cual le agradaría consultar cuáles son las mejores opciones.

Suponga que el propósito del usuario, situado en Los Palos Grandes, es almorzar con cinco personas desde las 2 p.m. hasta las 4 p.m. Estas condiciones conforman el ambiente inicial.

El usuario desea conseguir un restaurante cercano a su ubicación y entre las zonas de Chacao y Altamira (preferencias o dimensiones espaciales), pero también desea establecer prioridades en algunas características del restaurante como el tipo de comida y la categoría, y además indica que deben ser mínimas algunas características cualitativas como el tiempo de espera promedio para ser atendido y la cantidad de reclamos o malas referencias que posee el restaurante (preferencias o dimensiones no espaciales). A través de estas preferencias y la ubicación del usuario, la consulta retorna el conjunto de los mejores restaurantes que tienen disponibilidad para cinco personas de 2 a 4 p.m.

Sin embargo, surge un nuevo escenario, el individuo se encuentra en un vehículo y se ha movilizado dos cuadras hacia la plaza de La Castellana, esto origina un cambio en el ambiente inicial. Este cambio hace que algunos restaurantes recomendados situados en Los Palos Grandes deban ser descartados porque se encuentran más lejos, así como existen otros establecimientos en los alrededores de la plaza que ahora son una buena opción en cercanía.

Como el tiempo está transcurriendo, es posible que en pocos minutos existan restaurantes que hayan cambiado de disponibilidad debido a que otros clientes terminaron antes de comer, cancelaron una reservación o realizaron una nueva, lo cual genera otra alteración en el ambiente. Por lo tanto, si en esas recomendaciones, aparece un restaurante que ahora esté en su capacidad máxima o si no incluye un buen restaurante que pasó a estar disponible, se tendría un resultado equivocado.

Por estos motivos, es necesario refrescar la lista de recomendaciones dadas por la consulta sin que el tiempo de respuesta perjudique al usuario. Entonces, la nueva respuesta de la consulta debe ser el conjunto de los mejores restaurantes según los criterios previamente dados y la configuración actual del ambiente.

En base a lo anterior se define la problemática de este trabajo. Estos cambios de disponibilidad no están contemplados en los algoritmos que han sido desarrollados anteriormente para tratar el *skyline* espacial. El algoritmo *VCS*² [1] considera múltiples puntos de refe-

rencia y permite el desplazamiento de un punto a la vez, además que en su forma original no admite dimensiones no espaciales. El algoritmo *SSP* [8] reduce el número de puntos de referencia a sólo la ubicación del objetivo en movimiento, pero incorpora, como parte del problema, preferencias categóricas traducidas a lo que se denomina dimensiones no espaciales. Ambos algoritmos suponen que los objetos son estáticos, por lo que en su forma original no permiten gestionar la disponibilidad de los restaurantes del ejemplo anterior.

El propósito de este trabajo de grado es ofrecer una solución al problema del *skyline* espacial sobre datos dinámicos (*dynamic spatial skyline*, DSS), que consiste en la evaluación de consultas *skyline* espaciales basadas en múltiples dimensiones espaciales o puntos de referencia y en el cambio de ubicación de cualquiera de estos, sobre objetos con atributos adicionales a sus coordenadas geográficas. Además uno de los atributos, es una categoría altamente cambiante que condiciona la presencia del objeto como candidato al *skyline*, prohibiéndolo o habilitando según sea el caso que se origine y, el resto de los atributos, son atributos no espaciales a ser empleados en las preferencias de la consulta.

En este trabajo de grado, se proponen los algoritmos 2B2S, IB2S, VC2S+ y CD2S como distintas soluciones al problema DSS.

El algoritmo 2B2S, utiliza una estructura para almacenar el último valor de la categoría binaria cambiante de cada uno de los objetos. En base a los objetos habilitados para el *skyline*, evalúa la consulta utilizando el algoritmo BNL [3], este proceso lo realiza con cada alteración que ocurra en el ambiente. Este algoritmo aunque es ineficiente, fue implementado como referencia para validar que los demás algoritmos retornen el conjunto *skyline* correcto.

El algoritmo IB2S es una versión mejorada del anterior, porque actualiza el resultado del *skyline* en base a los cambios de estado del objeto presentes en el nuevo ambiente.

El algoritmo VC2S+ es una adaptación del algoritmo *VCS²* [1] para incluir dimensiones no espaciales y permitir cambios en los objetos, implementado como referencia para la

comparación de las demás soluciones. A diferencia de los otros dos algoritmos, VC2S+ actualiza el conjunto *skyline* con el cambio de posición del punto de referencia que se desplazó (en la mayoría de los casos), pero en presencia de cambios de disponibilidad en el ambiente, el algoritmo procesa los cambios de estado generando una nueva representación con los objetos habilitados y calcula el nuevo *skyline*.

El algoritmo CD2S es una solución mejorada porque actualiza el *skyline* con cualquiera de las transformaciones generadas en el ambiente. Adicionalmente, este algoritmo separa el *skyline* en cuatro grupos según las características de los objetos. De esta manera, CD2S conoce las porciones del *skyline* que no se ven afectadas por el desplazamiento del punto de referencia y las mantiene intactas, además que separa el proceso de actualización del *skyline* según estos grupos para reducir el número de comparaciones.

Los algoritmos implementados fueron comparados y analizados mediante seis experimentos diseñados para evaluar el desempeño de estos bajo las consultas DSSQ, en distintas condiciones iniciales y subsecuentes transformaciones del ambiente.

El trabajo consta de seis capítulos. En el Capítulo I, se delimita y define formalmente el problema DSS. En el Capítulo II se estudian los antecedentes y trabajos relacionados en el área. En el Capítulo III, se establece cómo fue representado el problema y se describen los cuatro algoritmos propuestos e implementados. El Capítulo IV comprende el diseño del estudio experimental realizado con datos sintéticos. En el Capítulo V, se explican los seis experimentos ejecutados, los resultados obtenidos y el análisis de cada uno. Finalmente, en el Capítulo VI, se presentan las conclusiones del trabajo elaborado, las recomendaciones y propuestas para trabajos futuros.

CAPÍTULO I

PLANTEAMIENTO DEL PROBLEMA

Las consultas *skyline* son generalmente aplicadas en problemas de planificación y toma de decisiones que involucran más de una variable [3]. Estas consultas permiten seleccionar, dentro de un gran conjunto de datos, aquellos objetos que mejor se ajusten a uno o múltiples criterios de preferencias definido. El criterio de preferencia se expresa indicando la variable involucrada en el problema cuyos valores se deseen maximizar o minimizar.

Las consultas *skyline* han evolucionado para manipular datos espaciales y agregar preferencias sobre las distancias de esos datos espaciales a un conjunto de puntos. Estas consultas se denominan consultas *skyline* espaciales (SSQ) [7].

Estas consultas tienen diversas aplicaciones. Pueden utilizarse en la logística de venta de entradas, asignación de puestos de estacionamientos y asistencia en festivales que alberguen multitudes como el *Rock in Rio*, *Pinkpop Festival*, *Lollapalooza* o *Glastonbury Festival* donde acuden más de 100.000 personas anualmente. También pueden emplearse con el mismo propósito, en eventos deportivos como la Liga Venezolana de Béisbol Profesional, la Copa América o los catalogados por *National Geographic* como los más importantes: las 24 Horas de *Le Mans*, los Juegos Olímpicos, el Mundial de Fútbol, el *Super Bowl*, las finales de la NBA o el torneo de Wimbledon los cuales albergan miles de ciudadanos y turistas de diferentes regiones y países [9].

Similarmente pueden ser aplicadas en situaciones cotidianas, como en la búsqueda de los mejores puestos para estacionar un vehículo en el campus de la Universidad Simón Bolívar, que estén cercanos al edificio donde el estudiante asistirá a la próxima clase y

a su ubicación actual; en aeropuertos, como el Aeropuerto Internacional de Maiquetía Simón Bolívar, con múltiples entradas peatonales que dependen de la aerolínea a la cual el usuario desea dirigirse, por lo que el usuario preferirá un puesto cercano a estas entradas; en parques temáticos y *resorts* como los existentes en Walt Disney World donde diariamente acuden miles de personas, por lo que necesariamente tienen que disponer de varios estacionamientos para los vehículos de los clientes.

También pueden ser empleadas en espacios abiertos, para facilitarle al usuario la escogencia de un puesto en las calles de zonas concurridas como el Centro de Caracas, con la necesidad de conseguir un puesto cercano al sitio donde el usuario realizará sus diligencias. En ciudades europeas como Roma o Barcelona, son pocos los estacionamientos techados existentes en el centro de la ciudad para la densidad de la población, debido a la preservación del casco histórico, por lo que existen sectores en las calles y avenidas de zonas residenciales y comerciales, con distintos horarios y costos por horas para estacionar. Por lo tanto, sería de gran ayuda una solución que le ahorre tiempo al conductor del vehículo al conseguir los mejores puestos según sus preferencias, entre otras posibles aplicaciones.

Estas aplicaciones van más allá del alcance de las consultas *skyline* Espaciales convencionales, porque suponen datos dinámicos ya que estos varían en el tiempo. En el caso del ejemplo de la escogencia de un puesto, éste puede cambiar de disponible a ocupado o viceversa, no se mantiene un mismo valor. Es por esto que en este trabajo de grado, se introduce un nuevo tipo de consultas, las consultas *skyline* espaciales sobre datos dinámicos (DSSQ).

En la sección 1.1 se presenta un ejemplo detallado como motivación al uso de consultas DSSQ para introducir intuitivamente el problema a resolver, en la sección 1.2 se presentan las definiciones básicas del *skyline* y en la sección 1.3 se define formalmente el problema.

1.1 Motivación

Supóngase un sistema instalado en los vehículos capaz de sugerir puestos disponibles en un estacionamiento. Un usuario del estacionamiento puede estar interesado en encontrar

un puesto disponible que tenga ciertas características, por ejemplo, que sea mínima la distancia a la salida peatonal y/o a alguna de las salidas del estacionamiento y que mientras se desplaza en su vehículo, la distancia al puesto destino sea mínima. Estas preferencias constituyen las dimensiones espaciales y, las posiciones de las salidas y el vehículo, forman los puntos de referencia de la consulta.

En este ejemplo, el movimiento del vehículo se traduce en una distancia dinámica, la cual variará para cada puesto con la nueva posición del vehículo.

Además, los puestos pueden tener dimensiones no espaciales tales como el tiempo de salida desde ellos. Por lo que el usuario preferirá un puesto que en las horas de mayor congestión el tiempo sea reducido.

Adicional a las dimensiones espaciales y no espaciales, los puestos de un estacionamiento, tienen un atributo variable que los definen como dato dinámico, el estado disponible u ocupado.

Un puesto disponible que cumpla con las características anteriores es considerado el mejor puesto para el usuario. Un puesto será mejor que otro, si posee un mejor valor en todas las características deseables.

En la Figura 1.1 se muestra una posible distribución de puestos de un estacionamiento y en la Tabla 1.1 los valores de los mismos para las características antes mencionadas. Formalmente estas características a ser consideradas en la escogencia de los mejores puestos son: la distancia en metros del vehículo al puesto $d(v, p)$, la distancia en metros del puesto a la salida peatonal $d(p, s_p)$, la distancia en metros del puesto a la salida del estacionamiento $d(p, s_e)$ y el tiempo de salida promedio en minutos $t(p, s_e)$.

Observe que el puesto 4 del **Sector 3, Fila B (3B-4)** es mejor que el puesto 2 del **Sector 3, Fila B (3B-2)**. Claramente, el puesto 3B-4 resaltado en color verde, posee la menor distancia al vehículo, entre todos los puestos del estacionamiento, pues su valor es $d(v, p) =$

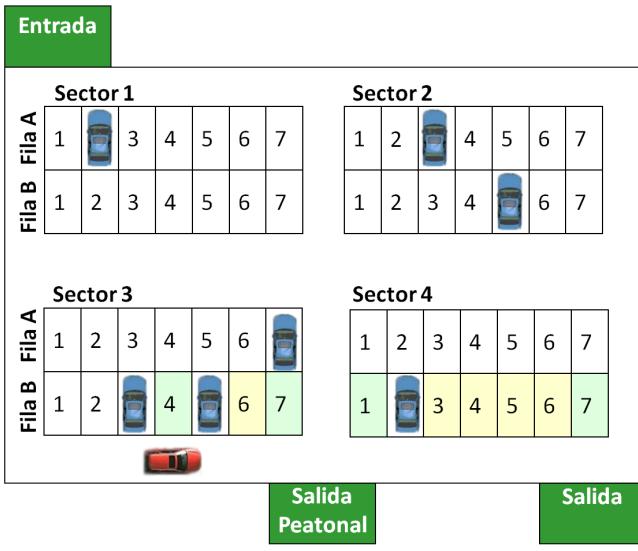


Figura 1.1: Ambiente inicial

Puesto	$d(v, p)$	$d(p, s_p)$	$d(p, s_e)$	$t(p, s_e)$	Sugerido
1A-*	>	>	>	>	No
2A-*	>	>	>	>	No
1B-*	>	>	>	>	No
2B-*	>	>	>	>	No
3A-*	>	>	>	>	No
4A-*	>	>	>	>	No
3B-1	3,0	7,5	14,5	5	No
3B-2	2,0	6,5	13,5	5	No
3B-4	0,4	4,5	11,5	4,5	Si
3B-6	2,0	2,5	9,5	4,0	Si
3B-7	3,0	1,5	8,5	4,0	Si
4B-1	5,0	1,5	6,5	3,0	Si
4B-3	7,0	3,5	4,5	2,8	Si
4B-4	8,0	4,5	3,5	2,8	Si
4B-5	9,0	5,5	2,5	2,8	Si
4B-6	10,0	6,5	1,5	2,5	Si
4B-7	11,0	7,5	0,5	2,2	Si

Tabla 1.1: Características de los datos

0,4 metros. Además, es mejor que el puesto 3B-2 en el resto de las características. Por lo tanto, el puesto 3B-4 estará en la respuesta del sistema porque no existe uno mejor que éste y, contrariamente, el puesto 3B-2 será descartado debido a que 3B-4 es mejor que él.

Por otra parte, pueden existir puestos que, aunque no cumplen con la menor distancia al vehículo, satisfacen otros atributos, lo que los hace incomparables entre sí por ser igualmente buenos. Los puestos 3B-7 y 4B-1 resaltados en verde, como son los puestos que están más cerca de la salida peatonal, tienen el mínimo valor para el atributo $d(p, s_p)$; y el 4B-7 resaltado en verde, es el que está más cerca, en tiempo y distancia, de la salida del estacionamiento así que tiene los menores valores en los atributos $d(p, s_e)$ y $t(p, s_e)$. Todos estos puestos resaltados en verde, satisfacen las preferencias deseadas por el usuario porque minimizan alguna característica, pero ninguno las satisface todas de manera simultánea. Véase que los puestos 3B-7 y 4B-1 están a igual distancia de la salida peatonal, pero uno está más cerca del vehículo y el otro de la salida del estacionamiento.

A diferencia de los puestos anteriores, el puesto 3B-6 y los puestos del 4B-3 al 4B-6, resaltados en amarillo, no poseen características sobresalientes (ni siquiera una que tenga el mínimo valor); sin embargo, también forman parte del conjunto de los mejores puestos.

Estos puestos, son mejores que los anteriores en las características restantes a las que minimizan esos puestos. Como ninguna característica tiene más importancia que la otra, todos estos puestos serán sugeridos por el sistema y será tarea del conductor escoger en cuál de los puestos sugeridos se estacionará.

El resto de los puestos no resaltados en la Figura 1.1, son descartados de las sugerencias del sistema, debido a que no son tan buenos como los anteriores. Basta con que un puesto tenga apenas una sola característica en desventaja con respecto a las características de los puestos sugeridos y ninguna más favorable, para dejar de ser interesante. Por ejemplo, el puesto 3A-6 es descartado por el 3B-6 porque está más lejos del vehículo, de la salida peatonal y de la salida del estacionamiento, y así ocurre con los demás puestos que en la Tabla 1.1, poseen valores en todas las características mayores al resto ($>$), estos puestos son los correspondientes a las **Filas A** y **B** en los **Sectores 1** y **2** y los que están en la **Fila A** de los **Sectores 3** y **4**.

En este ejemplo, los valores de las características $d(p, s_e)$ y $t(p, s_e)$ están correlacionados, porque comúnmente los puestos más cercanos a la salida son los que tienen menor tiempo de salida, pero podría ocurrir que un puesto sea muy bueno sólo en su dimensión no espacial y que esté alejado de los puntos de referencia. Si el puesto 1B-A tuviese el valor $t(p, s_e) = 1,0$ entonces formaría parte del *skyline*.

Existen puestos que no están habilitados para el *skyline* porque están ocupados por otro vehículo, como los puestos 3B-3, 3B-5 y 4B-2. Estos puestos ni siquiera son considerados en el proceso de obtención de las sugerencias, de lo contrario se podría generar un resultado errado.

Una vez que un usuario estacione su vehículo, el puesto que seleccionó dejará de estar disponible. Por lo tanto, el sistema no sólo debe adaptarse a los cambios de posición del vehículo, sino también a los cambios en la disponibilidad de los puestos. En la Figura 1.2 se muestra una posible alteración del ambiente inicial y en la Tabla 1.2 los nuevos valores en las distintas características de los puestos.

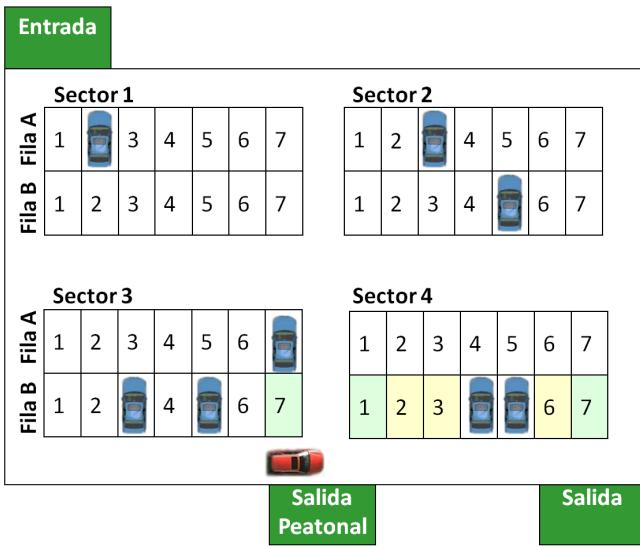


Figura 1.2: Ambiente transformado

Puesto	$d(v, p)$	$d(p, s_p)$	$d(p, s_e)$	$t(p, s_e)$	Sugerido
1A-*	>	>	>	>	No
2A-*	>	>	>	>	No
1B-*	>	>	>	>	No
2B-*	>	>	>	>	No
3A-*	>	>	>	>	No
4A-*	>	>	>	>	No
3B-1	6,0	7,5	14,5	5	No
3B-2	5,0	6,5	13,5	5	No
3B-4	3,0	4,5	11,5	4,5	No
3B-6	1,0	2,5	9,5	4,0	No
3B-7	0,5	1,5	8,5	4,0	Si
4B-1	2,0	1,5	6,5	3,0	Si
4B-2	3,0	3,5	4,5	2,8	Si
4B-3	4,0	3,5	4,5	2,8	Si
4B-6	7,0	6,5	1,5	2,5	Si
4B-7	8,0	7,5	0,5	2,2	Si

Tabla 1.2: Características de los datos

Debido a los cambios ocurridos, los puestos 4B-4 y 4B-5 son descartados de la lista de sugerencias porque fueron ocupados y, el puesto 4B-2 es incluido por encontrarse disponible y poseer características deseables.

Adicionalmente, se puede observar que el conjunto de puestos sugeridos cambió debido al desplazamiento del vehículo. Las distancias $d(v, p)$ tuvieron que ser calculadas nuevamente, y los puestos como el 3B-4 y 3B-6, que anteriormente eran buenos, ahora son descartados ya que pasaron a estar en desventaja en relación al puesto 3B-7.

En resumen, de acuerdo a las características deseadas por el usuario, no todos los puestos disponibles en el estacionamiento resultaron buenos: algunos estaban lejos del vehículo, de la salida peatonal y de la salida del estacionamiento. Por lo tanto, el sistema debe diferenciar cuáles son los mejores puestos entre todos los disponibles.

Para que el sistema conozca cuáles puestos están disponibles se puede emplear una red de sensores inalámbricos (WSN, por sus siglas en inglés). Un sensor puede colocarse abarcando uno o varios puestos y otro sensor en cada esquina de los pisos del estacionamiento. Los sensores se organizan y colaboran entre sí para enviar los datos recolectados a una o varias estaciones base. Los sensores colocados en los puestos del estacionamiento indican si el

puesto está disponible y dónde está ubicado. En cambio, los sensores colocados en las esquinas del piso del estacionamiento miden el número de vehículos por minuto para tener una muestra real del tráfico por piso y poder utilizarla al elaborar la recomendación.

Múltiples soluciones han sido planteadas para tratar problemas similares al presentado en este ejemplo de motivación. La mayoría de las soluciones desarrolladas utilizan WSN para resolver este problema, pero no están enfocadas en proveer al usuario recomendaciones y, mucho menos, incluyen los trabajos actuales que fusionan las WSN con el *skyline*.

Gran parte de estas soluciones se reducen al tema particular de estacionamientos, basándose únicamente en la tarea de conseguir puestos libres. En [10] se elaboró un sencillo sistema manejador de estacionamientos que utiliza LEDs para guiar al conductor hacia puestos disponibles sin realizar recomendaciones. Por otro lado, en [11] se presenta un sistema conformado por una WSN para recolectar información de los puestos, la cual periódicamente es transmitida y almacenada en una base de datos central y consultada para obtener puestos disponibles. En [12] se estudia la dificultad en la interacción con los sensores. En [13] es realizado un estudio de sensores con más profundidad, donde fue propuesta una red híbrida de sensores magnéticos y ultrasónicos, y algoritmos para la detección de los puestos disponibles. Una solución más compleja es propuesta en [14], en la cual es desarrollada una aplicación que muestra, en tiempo real, el mapa del estacionamiento con todos los puestos, distinguiendo los disponibles y permitiendo la reserva de puestos. En resumen, estas propuestas están enfocadas únicamente a verificar la disponibilidad de los puestos de un estacionamiento, sin dar la posibilidad de ser aplicadas a problemas más generales o a establecer preferencias sobre los mismos datos.

Otras soluciones enfocadas más aún a este ejemplo han sido implementadas. En [15], se presenta un sistema multi-agentes, localizador de puestos dentro de un campus universitario, que se basa en la comunicación de diversos dispositivos que colaboran entre sí, para que los agentes puedan proporcionar una lista de puestos ordenada. El orden de la lista es dado según la adaptación al usuario en términos de cercanía del puesto al lugar de trabajo o considerando las tarifas de estacionamiento, pero son valores estáticos que no

cambian según la posición del vehículo y la adaptación es realizada a partir del perfil que previamente el usuario creó en el sistema. Análogamente, en [16] se implementa un sistema que le permite al usuario, desde la entrada del estacionamiento, seleccionar el patrón de su salida peatonal destino preferida. Además, puede escoger el puesto disponible más cercano a esa salida mediante sensores ultrasónicos y, en base al algoritmo de búsqueda A*, se calcula el camino mínimo.

No obstante, estos últimos dos sistemas realizan la sugerencia al momento en que el usuario accede al estacionamiento, sin tomar en cuenta, la posición, ni el desplazamiento del vehículo. Además, estos sistemas permiten establecer como máximo dos criterios de selección tales como encontrar un puesto con la menor distancia al lugar de destino o con la menor tarifa posible, ambas características calculadas y almacenadas previamente.

Es de suma importancia destacar que, a diferencia de estas soluciones, el problema propuesto en este trabajo de investigación no se encuentra atado, en lo absoluto, a este ejemplo de motivación. Este ejemplo, se utilizará a lo largo del trabajo por ser sencillo y fácil de recordar.

Soluciones más generales pueden ser utilizadas. El método de selección de los mejores puestos puede realizarse mediante consultas *skyline* [3], las cuales permiten múltiples preferencias. Los puestos sugeridos y resaltados en las Figuras 1.1 y 1.2 son aquellos disponibles, que pueden ser identificados por una consulta *skyline* en donde se deseé la minimización de los atributos $d(v, p)$, $d(p, s_p)$, $d(p, s_e)$ y $t(p, s_e)$, simultáneamente.

Las consultas *skyline* tradicionales asumen que los atributos son estáticos y están almacenados en una base de datos [3, 5, 17–19], es decir que aplicadas a este ejemplo, los valores de los atributos $d(v, p)$, $d(p, s_p)$, $d(p, s_e)$ y $t(p, s_e)$ se mantienen en el tiempo. Sin embargo, en este ejemplo, la característica que indica qué tan cerca está el puesto del vehículo, $d(v, p)$, es una función que varía según la posición del vehículo, por lo que continuamente debe ser calculada durante el desplazamiento del vehículo y, en consecuencia, se tendría que realizar una nueva consulta *skyline* por cada variación de esta función.

El cálculo del *skyline* es costoso por la cantidad de comparaciones que deben ser realizadas entre todos los puestos, para escoger los mejores. La complejidad del peor caso es de $O(|P|^2)$ al utilizar el algoritmo Block Nested Loop, donde P es el conjunto de puestos [19]. Evidentemente, en presencia de funciones dinámicas es más costoso aún, porque además de estas comparaciones, se deben examinar previamente los valores generados por las funciones dinámicas, con respecto a cada uno de los puntos de referencia. En este ejemplo, los puntos de referencia utilizados para calcular las funciones $d(v, p)$, $d(p, s_p)$, $d(p, s_e)$ y $t(p, s_e)$ son: la posición del vehículo y las posiciones de las salidas. Realizando búsqueda exhaustiva, el espacio incrementaría a $O(|P|^2|Q|)$ donde Q es el conjunto de los puntos de referencia [7].

El *skyline* puede adaptarse a estos cambios de una mejor forma mediante las consultas *skyline* espaciales (*spatial skyline queries*, SSQ) [7]. Las SSQ están definidas por las características estáticas utilizadas en consultas *skyline* tradicionales y, adicionalmente, por características espaciales. Las características espaciales tienen una única interpretación geométrica correspondiente en el *skyline* espacial: miden la distancia euclíadiana del objeto a un punto de referencia que haya sido expresado en la consulta. En este ejemplo, la función $d(v, p)$ es una característica espacial debido a que su valor depende de la ubicación del vehículo para ese instante y, por lo tanto, es calculada dinámicamente basándose en la consulta del usuario, la cual proporciona las coordenadas del vehículo.

Las SSQ aprovechan las propiedades geométricas del espacio dado por el problema para reducir el espacio de búsqueda a $O(|S|^2|C| + \sqrt{|P|})$, donde S es el conjunto *skyline* y C es el conjunto de vértices del casco convexo formado por el conjunto de puntos de referencia Q . Estas técnicas geométricas utilizadas en [7] permiten podar comparaciones innecesarias para identificar el conjunto *skyline*, por lo que disminuye el costo de las comparaciones, evitando la revisión exhaustiva de cada objeto, en este caso, puesto.

Para evitar la restricción de las funciones de distancias euclidianas, se podría utilizar un enfoque más general que permite la adaptación de consultas *skyline* a atributos dinámicos, el *metric skyline* [20], el cual también permite podar el espacio de búsqueda.

Aún cuando es deseable utilizar estas consultas, porque admiten características espaciales en los datos, debido a la cantidad de variables involucradas y datos cambiantes, estas consultas en su forma tradicional no son apropiadas. No se adaptan a los cambios en el tiempo que puedan presentar los objetos, como por ejemplo, el atributo correspondiente al estatus de disponibilidad de un puesto. Por lo tanto, es conveniente extender las consultas SSQ, para proporcionar nuevas técnicas que permitan manipular datos espaciales y altamente cambiantes a la vez. Dicha extensión, en este trabajo de grado, se denominó consultas *skyline* espaciales sobre datos dinámicos (*dynamic spatial skyline queries*, DSSQ).

Por lo tanto, en este ejemplo, la gran cantidad de datos dinámicos recolectados por los sensores deben ser transmitidos constantemente y en forma organizada para ejecutar las consultas DSSQ y así evaluar y formar las sugerencias con la escogencia de puestos que más le convengan al usuario.

La construcción de sugerencias de puestos es sólo un caso particular de las diversas aplicaciones que tiene este problema. Incluso, por la naturaleza del *skyline*, como se dijo inicialmente, estas consultas pueden emplearse en la resolución de múltiples casos de gestión de recursos, recomendación y toma de decisiones.

En este trabajo, se proponen cuatro algoritmos que utilizan distintas técnicas para evaluar consultas DSSQ, las cuales permiten manejar preferencias sobre datos espaciales y cambiantes con el objetivo de resolver el problema *skyline* espacial dinámico (DSS, por sus siglas en inglés). Estos algoritmos son 2B2S, IB2S, VC2S+ y CD2S, los cuales son descritos en el Capítulo III.

La mejor técnica debe ser capaz de ofrecer un *skyline* adaptable al dinamismo de las características de los datos y de utilizar las propiedades espaciales de los objetos para podar el espacio de búsqueda con la finalidad de evitar búsquedas exhaustivas.

Asimismo, debido a que el cálculo del *skyline* puede ser costoso, sería de gran ayuda para el usuario contar con resultados parciales mientras se construye toda la respuesta de la

consulta, a la vez se desea evitar el cómputo de todo el conjunto *skyline* con cada alteración del ambiente, para así reducir el tiempo de evaluación.

1.2 Definiciones preliminares

Antes de introducir la definición formal del problema DSS, es necesario definir formalmente las consultas *skyline* en su forma convencional.

Las consultas *skyline* retornan los mejores objetos, según el orden parcial inducido por los criterios de preferencia ingresados por el usuario. Estos objetos conforman el conjunto *skyline*.

La forma de los criterios de preferencias empleados en estas consultas, son descritas en la Definición 1. El conjunto de los datos devueltos por estas consultas según estos criterios de preferencia se describen en la Definición 2. Además se definen el concepto de dominancia, de objetos incomparables y la pertenencia al conjunto *skyline* en las Definiciones 3, 4 y 5.

Sean $D = \{d_1, d_2, \dots, d_n\}$ el conjunto de datos u objetos y $A = \{a_1, a_2, \dots, a_m\}$ el conjunto de atributos que caracterizan los objetos pertenecientes a D . Se define:

Definición 1 (CRITERIO DE PREFERENCIA SKYLINE). *Un criterio de preferencia skyline p , es un **par ordenado** de la forma $p = \langle a, op \rangle$ compuesto por un atributo a perteneciente al conjunto A y una directiva op correspondiente a la operación de **minimización, maximización o agrupación** sobre el atributo a .*

El conjunto de estos criterios de preferencias C , con $C = \{p_1, \dots, p_k\}$ y k el número de preferencias definidas por el usuario, C induce un orden parcial \preceq sobre los objetos en D (D, \preceq). Se denota como A_C a los atributos en A para los cuales se definió algún criterio de preferencia en C .

Definición 2 (CONJUNTO SKYLINE). *El skyline S , con $S \subseteq D$, es el conjunto de **objetos no dominados o interesantes** que mejor se ajustan al conjunto de criterios de*

preferencia C que resultan del orden parcial (D, \preceq) inducido por C . El conjunto S se dice que domina los elementos restantes en $D - S$ porque esta formado por los maximales del orden parcial.

Definición 3 (DOMINANCIA SKYLINE). Sean x y y dos objetos, tal que $x, y \in D$, se dice que x **domina a** y , si y precede a x ($y \prec x$) en el orden parcial establecido por C . Es decir, y es dominado por x , si y sólo si, x es mejor o igual según los criterios de preferencia en C y es estrictamente mejor en al menos un criterio de preferencia. Formalmente:

Sean $x \in D$, $y \in D$, se cumple que x **domina a** y ($y \prec x$) si:

$$\star (\forall a_i \mid a_i \in A_C : y.a_i \preceq x.a_i) \wedge (\exists a_i \mid a_i \in A_C : y.a_i \prec x.a_i)$$

Donde: $x.a_i$ es el valor del atributo a_i para el objeto x .

Definición 4 (OBJETOS INCOMPARABLES). Sean los objetos x y y , tal que $x, y \in D$, se dice que x y y son **incomparables** al ocurrir uno de estos dos casos:

$$\begin{aligned} \star & (\exists a_i \mid a_i \in A_C : x.a_i \prec y.a_i) \wedge (\exists a_i \mid a_i \in A_C : y.a_i \prec x.a_i) \\ \star & (\forall a_i \mid a_i \in A_C : x.a_i = y.a_i) \end{aligned}$$

Definición 5 (PERTENENCIA AL CONJUNTO SKYLINE). Sea x un objeto, con $x \in D$, x pertenece al **conjunto skyline** ($x \in S$), si no está dominado por otro, es decir, si se satisface que:

$$\star \exists y \mid y \in (D - \{x\}) : x \prec y$$

Todos los objetos interesantes o no dominados por el resto del conjunto de datos, son incomparables entre sí y forman el skyline.

Las consultas *skyline* espaciales (*spatial skyline queries*, SSQ) [7], son más complejas. Éstas difieren del *skyline* tradicional en que los objetos son puntos de un conjunto P de un espacio

d-dimensional, R^d , en donde se define una función de distancia $dist$ denominada métrica de distancia, la cual es utilizada en las preferencias dadas por el usuario. Las preferencias se definen a partir de la métrica $dist$, como la minimización de las distancias de los objetos al conjunto de puntos de referencia Q (definidos en R^d) ingresados también por el usuario.

Sea $Q = \{q_1, \dots, q_k\}$ el conjunto de puntos de referencia. El *skyline* espacial está formado por el conjunto de objetos interesantes que mejor se ajustan a Q , según la métrica de distancia dada. Los mejores objetos son los que más cerca están de los elementos en Q , es decir, los que poseen menor valor en el rango de la métrica de distancia.

Las propiedades de la métrica de distancia son establecidas en la Definición 6, para la cual se redefine el criterio de preferencia en estas consultas en la Definición 7. En las Definiciones 8 y 9 se describe la dominancia espacial y la pertenencia al *skyline* espacial.

Definición 6 (MÉTRICA DE DISTANCIA EN EL SKYLINE ESPACIAL [20]). *Sea $dist(\cdot, \cdot)$ una función métrica de distancia definida en un espacio **d-dimensional** R^d , esta función satisface las siguientes propiedades:*

- ★ $(\forall x, y \mid x \in R^d \wedge y \in R^d : dist(x, y) \geq 0),$
- ★ $(\forall x, y \mid x \in R^d \wedge y \in R^d : dist(x, y) = 0 \Leftrightarrow x = y),$
- ★ $(\forall x, y \mid x \in R^d \wedge y \in R^d : dist(x, y) = dist(y, x)),$
- ★ $(\forall x, y, z \mid x \in R^d \wedge y \in R^d \wedge z \in R^d : dist(x, z) \leq dist(x, y) + dist(y, z))$ (*Desigualdad triangular*).

En el ejemplo de motivación, el espacio donde se encuentran los objetos es bidimensional (R^2) y la métrica de distancia utilizada es la distancia Euclidiana.

Definición 7 (CRITERIO DE PREFERENCIA SKYLINE ESPACIAL). *Un criterio de preferencia skyline espacial es de la forma $\langle dist(\cdot, q_j), \min \rangle$ y está compuesto por la métrica de distancia $dist$ a ser evaluada para cada uno de los objetos, el punto de referencia q_j*

perteneciente al conjunto Q y la directiva \min , la cual indica la **minimización** de esta distancia.

Definición 8 (DOMINANCIA SKYLINE ESPACIAL [7]). *Sean x y y dos puntos, con $x, y \in P$, x **domina espacialmente a y** , con respecto a Q , si para cada $q_i \in Q$, x se encuentra más cerca que y . Es decir:*

- * $(\forall q_i \mid q_i \in Q : dist(x, q_i) \leq dist(y, q_i)) \wedge (\exists q_j \mid q_j \in Q : dist(x, q_j) < dist(y, q_j))$.

Definición 9 (PERTENENCIA AL CONJUNTO SKYLINE ESPACIAL [7]). *Sea x un objeto con $x \in P$, x **pertenece al conjunto skyline espacial** si no está dominado espacialmente por otro, es decir, si satisface que:*

- * $(\forall y \mid y \in P \wedge y \neq x : (\exists q_i \mid q_i \in Q : dist(x, q_i) \leq dist(y, q_i)))$.

Como se puede observar en las definiciones anteriores, las consultas *skyline* espaciales no incluyen preferencias sobre los atributos que puedan tener los objetos.

Una variante de estas consultas se describen en [8], donde los autores reducen los aspectos espaciales a objetos situados en un espacio bidimensional y a un sólo punto de referencia q , pero amplían el concepto de objetos a datos con múltiples atributos referentes a categorías para las cuales el usuario define relaciones de dominancia (órdenes totales sobre los valores de cada una de las categorías).

La forma del criterio de diferencia es definida para esta variación de SSQ en la Definición 10, así como también el concepto de dominancia es descrito en la Definición 11 y la pertenencia al *skyline* se establece de la misma manera que para las consultas *skyline* convencionales en la Definición 5.

Para un conjunto de objetos D , cada objeto $o \in D$ posee un identificador, una coordenada espacial $l = (x, y)$, y un valor para cada atributo $a \in A$. Todos los atributos en A son categorías.

Definición 10 (CRITERIO DE PREFERENCIA PARA LA VARIANTE DE SSQ). *Tiene dos formas:*

Para la preferencia espacial sobre el punto de referencia dado q , es de la forma $\langle distEuc(\cdot, q), \min \rangle$ y está compuesto por: la distancia Euclídea ($distEuc$) a ser evaluada para cada una de las coordenadas de los objetos, el único punto de referencia q y la directiva \min , la cual indica la **minimización** de esta distancia.

Para las demás preferencias, es de la forma $\langle a, r \rangle$ compuesto por un atributo a perteneciente al conjunto A y la **relación de dominancia** r definida por el usuario para esa categoría.

Definición 11 (DOMINANCIA PARA LA VARIANTE DE SSQ [8]). *Sean x y y dos objetos, tal que $x, y \in D$, se dice que x **domina** a y , si y precede a x ($y \prec x$), es decir si x es mejor o igual a y en términos de todos los atributos y mejor que y en al menos uno. Formalmente:*

*Sean $x \in D$, $y \in D$, se cumple que x **domina** a y ($y \prec x$) si:*

$$\star (\forall a_i \mid a_i \in A^+ : y.a_i \preceq x.a_i) \wedge (\exists a_i \mid a_i \in A^+ : y.a_i \prec x.a_i)$$

Donde: A^+ es la unión de los atributos de categorías con el atributo de ubicación ($A^+ = A \cup \{l\}$) y $x.a_i$ es el valor del atributo a_i para el objeto x .

Esta variación de las consultas SSQ, aunque considera que el objeto puede tener otros atributos diferentes a su posición, se limita a un punto de referencia, a coordenadas bidimensionales, a la distancia Euclídea como métrica de distancia y tampoco considera un atributo altamente cambiante.

Una vez estudiados los conceptos relacionados al *skyline*, se introduce, en la sección 1.3, el problema formalmente.

1.3 Definición del problema

En el ejemplo de motivación, se planteó intuitivamente la necesidad de extender las consultas *skyline* espaciales para abarcar datos con atributos no espaciales estáticos y uno altamente cambiante que permita detectar los cambios de estado en los datos. Por esta razón, en este trabajo de grado se propone una extensión o adaptación de estas consultas, las **consultas skyline espaciales dinámicas (*dynamic spatial skyline queries, DSSQ*)**.

Las consultas DSSQ permiten preferencias espaciales, no espaciales y un requisito obligatorio, sobre un conjunto de datos D :

- ★ Las preferencias espaciales al igual que las SSQ, se definen a partir de una métrica de distancia $dist$ especificada (descrita en la Definición 6), como la minimización de las distancias de los objetos al conjunto de puntos de referencia Q , los cuales están definidos en un espacio d-dimensional (R^d).
- ★ Las preferencias no espaciales son similares a las existentes en las consultas *skyline* convencionales, se especifican sobre los atributos de los objetos. En presencia de atributos nominales o categorías, son traducidos, en un proceso previo, a atributos numéricos en base a las prioridades que defina el usuario.
- ★ El requisito obligatorio, se establece sobre el atributo altamente cambiante a_c de los objetos e indica la condición C_r que debe cumplir el objeto en base a los posibles valores de a_c para estar habilitado como candidato del *skyline*.

Los criterios de preferencia espaciales y no espaciales son descritos en la Definición 12, y el criterio obligatorio en la Definición 13.

Con respecto a los objetos evaluados por las consultas DSSQ, cada objeto $o \in D$ posee:

- ★ Un identificador o nombre que diferencia inequívocamente al objeto para que el resultado de la consulta sea entendible por el usuario.
- ★ Una ubicación u , la cual es un punto $p \in P$. Los puntos del conjunto P corresponden

al espacio d-dimensional, R^d .

- ★ Un valor para el atributo altamente cambiante a_c .
- ★ Un valor para cada atributo estático $a \in A - \{a_c\}$.

La dominancia para estos objetos es establecida en la Definición 14. Los mejores objetos son los que cumplen con la condición C_r , están más cerca de los elementos en Q y tienen los mejores valores según las preferencias no espaciales. La pertenencia al *skyline*, se describe en la Definición 15.

Definición 12 (CRITERIO DE PREFERENCIA PARA DSSQ). *Tiene dos formas:*

Para las preferencias espaciales, es de la forma $\langle dist(\cdot, q), \min \rangle$ y está compuesto por: la métrica de distancia $dist$ a ser evaluada para cada una de las coordenadas de los objetos, el punto de referencia q perteneciente a Q y la directiva \min , la cual indica la **minimización** de esta distancia.

Para las preferencias no espaciales, es de la forma $\langle a, op \rangle$ compuesto por un atributo a perteneciente al conjunto $A - \{a_c\}$ y una directiva op correspondiente a la operación de **minimización o maximización** sobre el atributo a .

Definición 13 (CRITERIO OBLIGATORIO PARA DSSQ). *Indica el **requisito** que debe cumplir un objeto para poder ser un candidato del skyline, es de la forma $C_r = \langle a_c, V \rangle$ y está compuesto por: el atributo altamente cambiante $a_c \in A$ y el conjunto de valores V del atributo a_c para los cuales están **habilitados** los objetos para el skyline.*

Definición 14 (DOMINANCIA PARA DSSQ). *Sean x y y dos objetos, tal que $x, y \in D$ y $x.a_c \in V$, se dice que x **domina** a y , si y precede a x ($y \prec x$), es decir si x es mejor o igual a y en términos de todos los atributos y distancias y mejor que y en al menos uno.*

*Formalmente, sean $x, y \in D$ y $x.a_c \in V$, se cumple que x **domina** a y ($y \prec x$) si:*

- * $y.a_c \in V \implies ((\forall q_i \mid q_i \in Q : dist(x, q_i) \leq dist(y, q_i)) \wedge (\forall a_i \mid a_i \in (A - \{a_c\}) : y.a_i \preceq x.a_i) \wedge ((\exists q_j \mid q_j \in Q : dist(x, q_j) < dist(y, q_j)) \vee (\exists a_i \mid a_i \in (A - \{a_c\}) : y.a_i \prec x.a_i)))$
- * $y.a_c \notin V \implies True$

Donde: $x.a_i$ es el valor del atributo a_i para el objeto x .

Definición 15 (PERTENENCIA AL CONJUNTO SKYLINE PARA DSSQ). *Sea x un objeto, con $x \in D$ y $x.a_c \in V$, x **pertenece al conjunto skyline** ($x \in S$), si no está dominado por otro, es decir, si se satisface que:*

- * $\lceil(\exists y \mid y \in (D - \{x\}) : x \prec y)$

En relación a lo anterior, se define el problema **skyline espacial sobre datos dinámicos** (**dynamic spatial skyline**, DSS), el cual consiste en la evaluación de las consultas DSSQ en ambientes dinámicos bien sea por nuevos valores del atributo cambiante para los objetos de entrada o, por el cambio de posición de alguno de los puntos de referencia establecidos en la consulta.

Para resolver el problema planteado, se estableció como objetivo general, diseñar e implementar una solución al problema de evaluación progresiva de consultas *skyline* espaciales en ambientes dinámicos. La evaluación progresiva en el *skyline*, consiste en el que el algoritmo de evaluación debe ser capaz de arrojar resultados iniciales sin que se haya terminado de calcular el conjunto *skyline* completo.

El objetivo general se descompone en los siguientes objetivos específicos:

- * Diseñar técnicas de evaluación progresiva para consultas *skyline* espaciales.
- * Desarrollar un algoritmo de evaluación progresiva de consultas *skyline* espaciales sobre datos cambiantes con mecanismos para la actualización del conjunto *skyline*.
- * Realizar un estudio experimental del desempeño de la solución propuesta.

CAPÍTULO II

EL *SKYLINE*, TRABAJOS RELACIONADOS

El *skyline* es un importante paradigma utilizado en la resolución de problemas de planificación y toma de decisiones (multi-criterio o multi-dimensional). El primer algoritmo para encontrar todos los *maximales*¹ de un conjunto de vectores fue propuesto en 1975 [4]. Se basaba en una estrategia *divide and conquer* y tenía un alto costo de procesamiento. Sin embargo, en el área de bases de datos, para ese momento, estaban centrados en la creación de sistemas manejadores de bases de datos relacionales y en la búsqueda de algoritmos eficientes de optimización y evaluación de consultas SQL [21], así que, mientras no se contara con buenas técnicas para el costoso procesamiento de los datos, era impensable la aplicación de problemas de ese estilo.

En 2001, luego de más de dos décadas, aparece la noción de consultas *skyline* para grandes bases de datos [3]. Desde entonces ha sido el eje de diversos trabajos. Debido a que la evaluación de consultas *skyline* es compleja, por la cantidad de comparaciones que deben ser realizadas entre los elementos, ha tomado importancia el diseño de algoritmos de evaluación en bases de datos relacionales que mejoren el tiempo de ejecución del *skyline*.

Cuando se tienen grandes cantidades de datos, mayor es la necesidad de un algoritmo de encontrar estos elementos. Los órdenes de complejidad en tiempo de algunos de los algoritmos más comunes, pueden ser observados en la Tabla 2.1. Todos los algoritmos referenciados dependen del número de dimensiones d del *skyline* y el tamaño n del conjunto de datos, así que mientras mayor sea el número de dimensiones y/o la cantidad de objetos, mayor será el tiempo que requerirá el algoritmo para evaluar el *skyline*.

¹En un conjunto parcialmente ordenado CPO, un maximal es un elemento para el cual no existe otro elemento mayor que lo siga en el orden parcial.

Algoritmo	Mejor caso	Peor caso
BNL [3]	$O(dn)$	$O(dn^2)$
SFS [22]	$O(dn + n\log n)$	$O(dn^2)$
LESS [2]	$O(dn)$	$O(dn^2)$

Tabla 2.1: Complejidad en tiempo de algunos algoritmos para el *skyline* [2]

El algoritmo BNL, *Block Nested Loops*, mantiene una ventana en memoria principal para colocar los posibles elementos *skyline*. Cada objeto o del conjunto de datos se compara (las d dimensiones) con todos los s_i que están en la ventana. Tal comparación tiene tres posibilidades: si o es dominado por algún s_i , se descarta o ; si o domina a algún s_i en la ventana, se elimina s_i y se agrega el nuevo objeto o ; y si o es incomparable con respecto a todos los s_i , se agrega y se continúa con el procedimiento. Evidentemente, el mejor caso se da cuando la ventana contiene un único elemento y el nuevo elemento a ser comparado reemplaza al de la ventana o es eliminado por él. Contrariamente, el peor caso es cuando la ventana es muy grande y no conforma una muestra representativa del conjunto *skyline*.

A diferencia del BNL, el algoritmo progresivo SFS, *Sort Filter Skyline*, realiza un ordenamiento topológico del conjunto de datos sobre las d dimensiones ($n\log n$), de forma tal que cada elemento que ingresa en la ventana es considerado *maximal*, lo cual reduce el número de comparaciones a ser realizadas. Sin embargo el peor caso sigue siendo $O(dn^2)$.

El algoritmo LESS, *Linear Elimination Sort for Skyline*, durante el primer paso del ordenamiento elimina parte de los objetos dominados, y en el último paso combina la ejecución con el filtro utilizado en SFS para eliminar los objetos restantes. La ventaja es que la eliminación temprana de los elementos es suficiente para afirmar que el costo de ordenamiento se reduce a $O(n)$, además que se realizan menos comparaciones.

Aún cuando, existen diversos algoritmos para las consultas *skyline* estáticas en bases de datos centralizadas, en general el peor caso tiene un orden de complejidad de $O(n^2)$. Intuitivamente, la evaluación de las consultas DSSQ es más compleja. El costo del cálculo de las distancias cada vez que existe un cambio en el ambiente y el costo de actualización

del conjunto *skyline* presentan un considerable incremento en el procesamiento de estas consultas. Sin embargo, son pocos los estudios acerca del cálculo del *skyline* en presencia de estos en ambientes dinámicos.

Las WSN conforman uno de los ejemplos más representativos de ambientes dinámicos, ya que manejan grandes cantidades de datos recolectados temporalmente por sensores que deben ser manipulados con técnicas que soporten ese dinamismo y sean escalables. La aplicación del *skyline* sobre WSN es prometedora debido a que resuelve problemas complejos que comúnmente surgen en sistemas de control, observación y monitorización de objetos.

La existencia de grandes cantidades de datos cuyos valores cambian en el tiempo es un objetivo retador para la evaluación de consultas en WSN [23]. Especialmente si se desea utilizar consultas *skyline* donde los datos suelen estar caracterizados por funciones o métricas de distancia con respecto a ciertos puntos de referencia. Es por esto que no son tan útiles los métodos empleados comúnmente en las consultas *skyline* tradicionales, tales como los algoritmos ya mencionados, índices calculados previamente [24] o modelos estadísticos de los datos. La mayor diferencia con estos trabajos está en que los datos son obtenidos al instante y no se encuentran almacenados.

Una característica probada del *skyline* espacial, en ambientes centralizados, es que en presencia de puntos de referencia móviles, los algoritmos que actualizan el conjunto *skyline* se comportan mejor que los que calculan de nuevo todo el conjunto. El tiempo promedio de respuesta disminuye en un 65 % [7]. Debido a esto, es necesaria la consideración de estas estrategias en ambientes dinámicos, para evitar el cálculo recurrente del *skyline* y sustituirlo por actualizaciones eficientes en el conjunto *skyline*. Por lo tanto, las estrategias estudiadas que se basan en la creación de particiones para obtener el *skyline* local de la región y la utilización de filtros para posteriormente transferir los objetos interesantes al *skyline* global, deben ser adaptadas para resolver este caso de estudio. Estas estrategias

sobre datos localizados y volátiles [25–27] resultan más eficientes que los algoritmos de *skyline* tradicionales. Permiten podar el espacio de búsqueda, a la vez que reducen la cantidad de datos a ser procesados y disminuyen la cantidad de veces que debe computarse completamente el conjunto *skyline* para un periodo de tiempo. Por consiguiente, aminoran el tiempo de evaluación y el costo de comunicación entre los dispositivos, en algunos casos, hasta un 91 % [27].

Por otro lado, es importante que la obtención del *skyline* espacial se realice de forma incremental o progresiva. De esta manera, el usuario puede disponer de resultados iniciales mientras se calcula el resto del *skyline*, lo cual es apreciado en problemas de toma de decisiones, ya que no siempre se necesita conocer *a priori* todas las posibilidades. Esta forma de obtener los elementos del *skyline*, es utilizada en la creación de particiones y filtros para reducir el costo generado por la transmisión y comparación de los datos [24, 28, 29]. En consecuencia, tomando en cuenta las necesidades y trabajos previos mencionados, se destaca la importancia de un mecanismo de evaluación progresiva de consultas DSSQ que permita puntos de referencia móviles y la adaptación de estas consultas a datos temporales sobre ambientes dinámicos.

Diversos estudios han sido realizados en el área de bases de datos para introducir el *skyline* como solución al problema de evaluación de consultas que expresan preferencias de usuarios. En las secciones 2.1, 2.2, 2.3, 2.4 y 2.5, son presentados los trabajos relacionados al estudio de las consultas *skyline*, del *skyline* espacial, otras consultas espaciales y la combinación de ambientes cambiantes con consultas *skyline* y *skyline* espaciales, métodos que han sido considerados como estudio inicial para dar una solución al problema DSS.

2.1 Consultas *skyline*

El conjunto *skyline* está conformado por elementos incomparables entre sí. Debido a su alto costo de procesamiento [19], diversos trabajos han sido realizados para calcularlo.

Inicialmente, las consultas *skyline* eran empleadas para resolver problemas donde se asumía que los datos estaban centralizados en una gran base de datos y los estudios [6, 22, 30–34]

iban dirigidos a proporcionar técnicas cada vez más eficientes para la optimización de estas consultas, como por ejemplo, utilización de índices y algoritmos de evaluación.

Por otra parte, para garantizar un menor tiempo de respuesta, fueron propuestos algoritmos que retornen el *skyline* de forma progresiva o en línea [5, 17, 18, 35]. Estos algoritmos permiten obtener los primeros resultados evitando que sea necesario esperar que todo el conjunto de datos sea examinado, y el resto de los resultados es devuelto de forma continua a medida que se van identificando los objetos *skyline*.

Todos estos trabajos se enfocan en bases de datos centralizadas. Sin embargo, debido al continuo crecimiento de la Web, el estudio del *skyline* se ha orientado a crear algoritmos distribuidos que procesen estas consultas [36] para realizar búsquedas eficientes sobre fuentes de datos descentralizadas.

Trabajos posteriores han sido elaborados sobre algoritmos progresivos en fuentes de datos distribuidas, donde el principal reto es reducir el costo de comunicación y de procesamiento del *skyline*. En [37] se consideró el problema de procesar el *skyline* en tiempo real sobre los elementos más recientes en un *data stream*.²

Así también, en [38–40] las consultas *skyline* son estudiadas en sistemas P2P (*peer to peer*) desarrollando algoritmos que procesan el *skyline* en paralelo sobre la red P2P.

Otros estudios se enfocaron en ampliar las consultas *skyline* para incluir atributos dinámicos como criterio de preferencia. En particular, se define el *dynamic skyline* en [35], donde los atributos de cada objeto son especificados mediante un conjunto funciones de dimensiones definidas por distancias Euclidianas.

²*Data stream* se refiere al flujo de una larga secuencia de datos transferida a una alta velocidad y de forma ininterrumpida.

2.2 Consultas *skyline* con datos espaciales

Existe una gran cantidad de variantes de consultas *skyline* aplicadas a datos espaciales. Las consultas más conocidas, son las consultas *skyline* espaciales (SSQ) [7], las cuales permiten que atributos dinámicos sean calculados a partir de puntos de referencia expresados en la consulta, por lo que el *skyline* retornará los objetos que mejor se aproximen a estos puntos. Esta aproximación a los puntos de referencia, se realiza por distancias Euclidianas. En ese trabajo, los autores proponen para puntos de referencia estáticos, el algoritmo B^2S^2 basado en *r-trees* y el algoritmo VS^2 basado en diagramas de *Voronoi* y en el *convex hull* que forman los puntos de referencia; el algoritmo VCS^2 , es propuesto para una actualización continua del *skyline* para puntos móviles, donde en los casos que no puede actualizar el conjunto *skyline* emplea el algoritmo VS^2 . Adicionalmente, muestran cómo esos algoritmos pueden ser modificados para una variación de SSQ que incluya atributos no espaciales, indicando los cambios que deben ser realizados en los algoritmos.

A partir de las consultas SSQ, en [41] se trata el problema del *skyline* espacial con datos masivos en espacios bidimensionales, en el cual demuestran que la primera versión del algoritmo VS^2 propuesto en [7] para distancias estáticas, fallaba en obtener algunos puntos del *skyline* para ciertos casos. Por lo que uno de los aportes de estos autores, es la corrección del algoritmo VS^2 y lo comparan con su algoritmo propuesto *ES*, el cual resulta mejor porque reduce las comparaciones de dominancia innecesarias.

Debido a que el algoritmo VS^2 fallaba en ciertos casos, los autores de las consultas SSQ, publican una segunda versión de los algoritmos VS^2 y VCS^2 en [1]. Ambos algoritmos fueron estudiados en este trabajo de grado para proponer el algoritmo VC2S+. El algoritmo *ES* no fue considerado, debido a que no tiene un mecanismo que actualice el *skyline* en presencia de puntos de referencias móviles.

Posteriormente, los mismos autores del algoritmo *ES* en [42], implementaron un algoritmo de aproximación para consultas continuas de movimiento *skyline* espaciales, el cual retorna un subconjunto del *skyline* espacial sin obtener todos los puntos. Estos puntos retornados,

son considerados los objetos más representativos del *skyline* porque maximizan la cantidad la cantidad de objetos dominados.

Otro tipo de consultas son definidas en [43], las consultas *multi-source skyline*, donde los atributos dinámicos representan distancias de los caminos mínimos encontrados en redes viales, con respecto a los puntos de referencia expresados en la consulta. Adicionalmente en [44], distintos tipos de consultas *skyline* con atributos espaciales son estudiadas, proponiendo algoritmos eficientes y escalables para estas consultas basados en índices.

Al mismo tiempo, una nueva variante de consultas *skyline* es propuestas en [45], donde los autores definen las consultas *location-dependent skyline* (LDSQ), las cuales consisten en un caso particular de las consultas SSQ para un solo punto de referencia, pero incluyen preferencias sobre atributos no espaciales. Además para el punto móvil, se introduce el concepto de *Valid Scope*, el cual consiste en una región espacial en la que el cambio de posición del punto de referencia producirá un resultado idéntico al anterior, por lo tanto, dentro de esa región la consulta no necesita ser relanzada para que los resultados sean válidos. Similar a estas consultas, en [8] se define una variación de las consultas SSQ, con preferencias sobre un punto de referencia (utilizando la distancia Euclídea) y las preferencias no espaciales corresponden a categorías, en las que el usuario define un orden total. Para estas consultas, los autores proponen un algoritmo que sea capaz de retornar al menos k buenos resultados, cuyo parámetro es especificado por el usuario.

Ambos estudios resultan interesantes para este trabajo de grado porque son incluidas preferencias no espaciales sobre los datos, pero no consideran atributos no espaciales altamente cambiantes. Por otro lado, es de gran relevancia contar con una región valida para la cual los datos se mantienen y la posibilidad de retornar al menos k objetos cuando la cardinalidad del *skyline* resulta pequeña.

A partir de las consultas LDSQ, en [46], son combinadas las preferencias espaciales y no espaciales del *skyline* con métodos de recuperación de información para realizar una recomendación basada en la localización del usuario y sus intereses.

Con respecto al estudio del desplazamiento del punto de referencia en el *skyline*, surgen las consultas *skyline* espaciales basadas en direcciones [47]. Las cuales retornan los objetos más cercanos alrededor del usuario (punto de referencia) en diferentes direcciones, sin tomar en cuenta atributos no espaciales, en un espacio bidimensional y bajo distancias Euclidianas, descartando los objetos en las direcciones no deseadas. Este trabajo es extendido en [48–50] para redes viales contemplando distancias y tiempo de viaje. Posteriormente en [51, 52], incluyen un ángulo máximo de desviación, para evitar que objetos fuera de esa dirección sean presentados como sugerencias y permiten el cálculo del *skyline* con atributos no espaciales estáticos. Los autores proponen dos algoritmos que actualizan el *skyline* de forma continua, considerando una dirección o múltiples y limitando el espacio de búsqueda. Al igual que el algoritmo CD2S propuesto en este trabajo de grado, estos algoritmos separan del *skyline* los objetos estáticos que se mantienen en él, que en este caso, son los que mejor se acercan a las preferencias no espaciales.

Consultas continuas similares, son presentadas en [53]. Estas son las consultas *skyline* basadas en distancias, las cuales son empleadas para redes viales y reciben como entrada preferencias no espaciales, un punto de referencia (el cual es el usuario en movimiento) y un camino a seguir. Estas consultas se dividen en dos, las CD-SQ que reciben un valor máximo de distancia y Cknn-SQ que reciben un k . Las primeras consultas, retornan los objetos en el rango de distancia especificado y las segundas, los k objetos más cercanos. Aunque el problema DSS no se basa en la especificación de un camino de movimiento, estos enfoques son relevantes para proporcionar otras variantes de conjuntos de objetos interesantes.

Tomando las consultas *skyline* basadas en localización, en [54, 55] son estudiadas las consultas *skyline* reversas (RSL), tanto para puntos de referencia móviles como para objetos en movimiento. Éstas, retornan el conjunto de objetos que incluye el punto de referencia como resultado de la consulta *skyline*. Un posible ejemplo de aplicación serviría para el dueño de un restaurante, el cual desea conocer todos los clientes que consideran su restaurante como el más económico y cercano. Para estas consultas, los autores presentan un método de evaluación continua que poda el espacio y actualiza el *skyline* incrementalmente.

Todos estos enfoques representan los objetos como vectores en el espacio o utilizan la propiedad geométrica Euclíadiana de los objetos para podar el espacio de búsqueda. En [56], se estudia el problema *skyline* espacial aplicado a la distancia Manhattan, que a diferencia de la Euclíadiana, puede adaptarse a las restricciones de las calles existentes en los mapas de ciudades. Los autores de ese trabajo de investigación, implementaron un algoritmo basado en el diagrama de Voronoi de los datos y el *convex hull* formado por los puntos de referencia con un orden de complejidad menor al algoritmo *ES*.

Sin embargo, en [20] se hace un estudio más general: se define el *metric skyline*, en el cual los atributos pueden ser medidos por diversas métricas (no sólo distancias euclidianas) y no restringe la forma de los objetos a vectores; pueden ser modeladas como polígonos o secuencias que mediante métricas adecuadas son transformados a vectores de atributos dinámicos. Años más tarde, en [57], es diseñado un algoritmo para la evaluación de estas consultas, utilizando índices *PM-tree*, el cual supera al algoritmo original basado en índices *M-tree*.

Aún cuando el *metric skyline* es más general, para este trabajo de investigación resultan de particular interés las SSQ por ajustarse más al problema planteado, además de ser las más conocidas, probadas y estudiadas en problemas espaciales, por lo que son estas consultas las escogidas para ser adaptadas a datos cambiantes y atributos no espaciales.

Por otro lado, las consultas sobre datos espaciales pueden ser un caso particular de las consultas *skyline* dinámicas, denominadas consultas *skyline* espaciales del vecino más cercano [58]. En estas consultas, el usuario tiene un conjunto de puntos de referencia para los cuales a través del algoritmo N^2S^2 , basado en algoritmo *Branch and Bound*, se obtienen los objetos más cercanos, utilizando como distancia de cada objeto el valor de la distancia de cada punto de referencia a su vecino más cercano. Lamentablemente, este enfoque no considera atributos no espaciales o puntos de referencia móviles.

Adicionalmente en [59, 60], se estudia una generalización del problema *skyline* espacial, denominado el problema *skyline* espacial general (GSSKY). Este problema, permite tener

en una consulta, varios tipos o categorías de puntos de referencia. Por ejemplo, las distintas estaciones de bus y supermercados, para las cuales se desea conseguir el mejor apartamento, en términos de cercanía a estos puntos. Pero, el mejor apartamento no necesariamente tiene que estar cercano a todos estos puntos, sino al menos a uno de cada tipo. Aunque el *skyline* espacial es un caso particular del SSKY, porque permite un solo punto de referencia para cada categoría o sólo una categoría para todos los puntos, este problema no resuelve la integración de preferencias no espaciales y atributos altamente cambiantes, además que el algoritmo Efficient GSSKY que procesa estas consultas, se enfoca en puntos de referencia con posiciones estáticas.

Un enfoque distinto al *skyline* con datos espaciales es estudiado en [61]. En el cual, son propuestas las consultas *farthest spatial skyline* (FSSQ) para un problema contrario, de maximización, en el cual la dualidad con las SSQ no se mantiene. En un espacio bidimensional, dado un conjunto de puntos P y otro de puntos de referencia Q estáticos, estas consultas tienen como objetivo devolver los puntos que se encuentran más lejos que el resto, en al menos un punto de referencia. Las FSSQ pueden ser aplicadas en la identificación de ubicaciones espaciales lejanas a lugares indeseables, como por ejemplo, desagradables o instalaciones de competidores comerciales. Para estas consultas, los autores desarrollan el algoritmo progresivo BBFS, basado en el diagrama de *Voronoi* de los datos y el *convex hull* que forman los puntos de referencia.

2.3 Otras consultas espaciales

Consultas distintas al *skyline* han sido estudiadas para datos espaciales. Las consultas más conocidas son las consultas del vecino más cercano (NN) [62, 63], las cuales retornan el punto más próximo a un punto de referencia, mediante búsquedas de profundidad o amplitud. Trabajos recientes [64], han estudiado la extensión de este problema a múltiples puntos de referencia, con las consultas *aggregate nearest neighbor* (ANN). Éstas permiten obtener los puntos más cercanos, con respecto a una función de distancia. No obstante, estas consultas obligan al usuario a elaborar su propia función de agregación para las distancias, lo cual no es práctico para un usuario final. En cambio, las consultas SSQ tienen la ventaja que el usuario sólo especifica una función monótona de distancia.

En [65], se propone un diagrama de *Voronoi* probabilístico para procesar las consultas de movimiento del vecino más cercano en datos inciertos y para un punto móvil (PMNN). En donde los autores proveen dos técnicas, el pre-computo del diagrama para el conjunto completo de datos y un enfoque incremental, en el que no es necesario construir todo el diagrama para responder a la consulta. Estas consultas, aunque tratan con datos inciertos y el desplazamiento de un punto de referencia, no incluyen preferencias no espaciales sobre los datos. De todas formas, es interesante el enfoque incremental para la construcción del diagrama de Voronoi de los datos, el cual podría estudiarse como otra posible representación de los datos para el algoritmo **VC2S+**, con respecto a la variabilidad del atributo altamente cambiante en el problema DSS.

Por otro lado en [66], es analizado el problema de evaluación de preferencias espaciales, numéricas y categóricas con la misma importancia o con distintas prioridades, sobre datos estáticos para *Preference SQL*, el cual es una extensión declarativa del estándar SQL. Otros estudios, se basan en las consultas *top-k* para establecer preferencias espaciales y no espaciales sobre datos espaciales en redes viales [67]. Estas consultas, a través de una función de agregación definida por el usuario según su propósito y mediante un algoritmo basado en índices *R-tree*, construyen un orden entre los objetos, retornando los k minimales o máximas con respecto a la función de agregación. Al mismo tiempo, en [68], son integradas las consultas *top-k* sobre palabras claves que describen los datos, para conseguir los k mejores objetos, en términos de la distancia al punto de referencia y su relevancia textual con respecto a las palabras claves, utilizando caminos de costo mínimo. Estos estudios, se encuentran muy alejados del problema planteado DSS, además de que están basados en objetos estáticos y no consideran consultas continuas consecuencia de cambios en el ambiente.

2.4 Consultas *skyline* en ambientes cambiantes

Con los avances en la tecnología, constantemente aparecen nuevos retos, y por la necesidad de automatizar las tareas surgen los sistemas de monitorización. Estas aplicaciones de control que manejan observaciones obtenidas y datos dinámicos masivos, han fomentado la creación de nuevas estrategias para procesar los datos emitidos por los sensores inalámbricos.

cos. El principal reto de dichas estrategias es mejorar la calidad de datos devueltos sin comprometer el costo de la consulta [69].

Las WSN son comúnmente utilizadas para monitorizar variables de control en aplicaciones militares, ambientales, médicas, en vigilancia de tráfico, en control industrial, en administración de taxis, en autos de policías, etc. Los sistemas de monitorización lidian continuamente con grandes cantidades de datos y por lo tanto, se deben desarrollar nuevas técnicas para gestionar estas WSN y buscar otras vías para soportar consultas más complicadas y costosas. Las técnicas deben ser eficientes en términos de energía y, además deben permitir hallar la respuesta requerida, pero reduciendo los costos de comunicación, de procesamiento, tiempos de respuesta y número de paquetes. Asimismo, se debe tener en cuenta las limitaciones de los sensores en almacenamiento y transmisión de datos y el retraso en las comunicaciones.

Recientemente se ha estudiado la aplicación de consultas *skyline* en WSN. La mayoría de los trabajos de investigación están dirigidos a reducir la información a ser transmitida entre los diversos dispositivos mediante filtros que calculan localmente el *skyline*. En [70] se expone un algoritmo para el procesamiento en una *in-network* de las consultas *skyline* que reduce el costo de comunicación y de distribución de la carga. En [71] se propone un algoritmo para soportar monitorizaciones *skyline* en una WSN, el cual emplea umbrales jerárquicos en los nodos para minimizar la cantidad de información transmitida. De la misma manera, en [72] es presentado el algoritmo *Sliding Window Skyline Monitoring Algorithm* (SWSMA) que descompone la consulta *skyline* y emplea filtros para reducir la cantidad de datos transferidos y ahorrar energía. Similarmente, en [72] es implementado un algoritmo, *Energy-Efficient Sliding Window Skyline Maintaining Algorithm* (EES), que emplea el filtro *Mapped Skyline*, el cual reside en el sensor y, calculando un *skyline* en cada nodo, filtra las tuplas que no tienen ninguna contribución al resultado final.

Referente al procesamiento y actualización del *skyline*, en [73] son expuestas técnicas eficientes que manejan el constante cambio en los datos recolectados por los sensores. Otros algoritmos de evaluación y mantenimiento del *skyline* en WSN para maximizar el

tiempo de vida de la red son desarrollados en [74]. Para encontrar un filtro, se buscan certificados de *skyline* locales y se realizan las comparaciones respecto a estos.

Para resolver el problema de monitorizar el ambiente en la prevención de incendios forestales, se presenta un algoritmo mejorado de *sliding windows* para la computación continua del *skyline* y así suprimir los datos recolectados por los sensores [75]. También, en [76] se propone un algoritmo para procesar consultas multi-atributos *Section Bit Based scheme*(SSB) que reduce la cantidad de mensajes innecesarios manteniendo la información parcial frecuentemente recogida por los sensores en ciertos nodos. Asimismo, en [77] se expone el impacto en la seguridad que causan ataques de colisiones e identificación de falsa información; para esto, han sido diseñados tres algoritmos para consultas multi-atributos en *Tiered Sensor Networks*, entre ellos el *secure skyline query*.

En cuanto a técnicas de procesamiento progresivo, en [28] dos algoritmos de evaluación (FDP y DDP) han sido propuestos. Estos encuentran los puntos del *skyline* de forma progresiva y, además, crean particiones del conjunto de datos en subconjuntos disjuntos y son examinados por separado. A medida que consigue los puntos *skyline* en el subconjunto, filtra aquellos no interesantes antes de ser transmitidos. De igual forma, en [29] se diseña el algoritmo *Dynamic Radius-Partition Based Algorithm (DRP)* que realiza particiones del conjunto de datos en subconjuntos disjuntos, y con cada evaluación progresiva local del *skyline* construye filtros globales para disminuir la transmisión de datos innecesarios. También se presenta el algoritmo *MSM* para mantener el *skyline* en un ambiente de *sliding windows* de forma incremental. Asimismo, en [24] se muestra el algoritmo *SkyTree*, el cual construye particiones dinámicas basadas en regiones que maximizan la dominancia, generadas por puntos pivote y calcula de forma progresiva y escalable el *skyline* mientras evita la comparación con todos los puntos.

Las soluciones antes mencionadas resuelven parte del problema presentado, ya que establecen técnicas para filtrar los datos a ser procesados, así como realizan un cálculo incremental del conjunto y proveen mecanismos de actualización del *skyline* en presencia de datos temporales.

El inconveniente con estas técnicas es que no permiten posiciones espaciales de los objetos, ni consideran el hecho que puedan estar en movimiento.

2.5 Consultas *skyline* no tradicionales en ambientes cambiantes

Consultas sobre datos distribuidos y ciertas nociones espaciales son descritas en [78], donde se estudia el *skyline* en redes *Mobile Ad-hoc Networks* (MANET), con estrategias de procesamiento, para reducir los datos a ser transferidos.

En [79] se abarca un problema más general que las *SSQ*; son estudiadas las consultas multi-dimensionales con características espaciales en clientes móviles para servicios basados en localización. Se hace uso de un índice (MR-tree) para autenticar consultas espaciales (*k Nearest Neighbors KNN* y *skyline* espaciales) y un protocolo de comunicación que, basándose en técnicas de sincronización, utiliza resultados de consultas previas para reducir el tamaño de la información enviada.

Posteriormente, es propuesto el algoritmo *Distributed Spatial Skyline* [27]; éste es el primer algoritmo distribuido para procesar consultas *skyline* espaciales, el cual reduce en gran medida la sobrecarga de comunicación con respecto a los algoritmos centralizados. La estrategia del algoritmo es buscar en paralelo conjuntos *skyline* locales creando particiones recursivas del espacio de búsqueda según las propiedades geométricas de los nodos y la topología. Similarmente, en [25] se trata el problema de consultas *skyline* basadas en localización y presentan el algoritmo *Ring-Skyline (RS)*. RS divide el área a monitorear en anillos, utiliza técnicas de reducción de consumo de energía en los sensores y progresivamente va retornando los resultados para reducir el tiempo de respuesta. Estos dos trabajos, solucionan parte del problema planteado, ya que ofrecen técnicas espaciales para procesar *skyline* aun cuando no estudien en profundidad la constante actualización del *skyline* causada por datos altamente dinámicos ni la inclusión de preferencias sobre atributos no espaciales.

Referente al manejo de objetos dinámicos, un trabajo interesante se ha realizado sobre la minimización del costo de comunicación en arquitecturas cliente-servidor [26], donde

el servidor mantiene continuamente el *skyline* de los objetos dinámicos. El *skyline* es calculado de forma exacta utilizando filtros, o de forma aproximada mediante técnicas de muestreo y, es mantenido según el algoritmo *Frequent Skyline Query over a Sliding Window* (FSQW) el cual reporta los objetos *skyline* frecuentes.

Por otro lado, en [80, 81], se estudia el *skyline* para objetos no estáticos. Tanto los objetos como el único punto de referencia se encuentran están movimiento, estos objetos pueden tener asociados atributos no espaciales bien sea dinámicos o estáticos. En ese estudio, es analizado el problema de consultas *skyline* Predictivas para este tipo de objetos, donde los autores implementan el prototipo PRISMO, basado en el algoritmo *Branch and Bound Skyline* (BBS) e índices *R-tree*. Este problema es similar al planteado DSS debido a que los datos cambian en el tiempo, aunque sólo se considera un punto de referencia.

Posteriormente en [82], es considerado el *Path Skyline* como una integración del procesamiento de objetos móviles y caminos de costo mínimo a partir de un punto de salida y uno de llegada, con el fin de obtener los objetos más cercanos.

Recientemente, se estudió en [83] el caso de rutas *skyline* para alimentar un sistema de recomendación móvil eficiente en el consumo de energía, el sistema es utilizado para sugerir a un conjunto de taxistas, las rutas que deben tomar para maximizar la cantidad de paradas y minimizar la distancia potencial a ser recorrida mientras encuentran a un pasajero. Además, una función de distancia *Potential Travel Distance (PTD)* es definida para evaluar cada secuencia de paradas candidata, que puede ser utilizada para podar el espacio de búsqueda. Esta función es utilizada por los algoritmos *LCP* y *SkyRoute* para encontrar las rutas recomendadas. Los mecanismos descritos anteriormente sirven de base para el problema planteado. Algunos de estos trabajos ofrecen técnicas para tratar con atributos dinámicos, otros admiten puntos en movimiento y varios permiten la actualización eficiente del *skyline* y la obtención progresiva de los elementos.

CAPÍTULO III

SOLUCIONES AL PROBLEMA DE DSS

En este capítulo se describen los cuatro algoritmos propuestos al problema de DSS: los algoritmos 2B2S e IB2S que representan dos soluciones ingenuas, el algoritmo VC2S+ que es una adaptación al problema utilizando los algoritmos VS^2 y VCS^2 [1] y el algoritmo CD2S que corresponde a la solución propuesta porque actualiza el *skyline* a la vez que reduce el número de comparaciones a realizar para obtener el conjunto.

En la sección 3.1, se explica cómo es la representación de las entradas de los algoritmos en general, se describe el algoritmo utilizado para la simulación de los cambios del ambiente, se muestra un ejemplo de estas simulaciones y se presentan brevemente los algoritmos en los que están basados los algoritmos 2B2S, IB2S, VC2S+ y CD2S. En la sección 3.2, se describe el algoritmo 2B2S y su variante el algoritmo IB2S, en la sección 3.3 se explica el algoritmo VC2S+ y en la sección 3.4, se presenta el algoritmo CD2S. Las características y diferencias entre los algoritmos son puntualizadas en el Apéndice D .

3.1 Representación del problema

La entrada a la consulta DSSQ es el conjunto de objetos espaciales a evaluar, los puntos de referencia estáticos y un punto de referencia móvil, los cuales conforman los datos necesarios para calcular el *skyline* espacial. Como entrada a estas consultas, también se pueden incluir atributos no espaciales, como lo son los valores de las dimensiones no espaciales, con los cuales se obtiene el *skyline* no espacial.

La salida o respuesta de la consulta será el conjunto de objetos recomendados por el *skyline* espacial y no espacial.

En el ejemplo de la sección 1.1 del capítulo I, los objetos espaciales se corresponden a los puestos de estacionamiento, los puntos de referencia estáticos a las salidas del estacionamiento u otros puntos de interés, el punto de referencia móvil al vehículo en desplazamiento, las dimensiones no espaciales, a características de los objetos adicionales a la distancia de los puntos de referencia, como lo es el tiempo de salida o el costo por hora del puesto.

Los datos de entrada, se utilizan para inicializar los algoritmos con la transmisión inicial que realizarían los sensores que detectan la ocupación de los objetos. Se utilizan archivos de texto plano, estos archivos son: **aConf** y **aPtosRef**. Adicionalmente, se utiliza el archivo **aDimNoEsp** para los valores de las dimensiones no espaciales de cada objeto.

El archivo **aConf**, contiene las dimensiones del plano de los objetos espaciales y la disponibilidad de cada uno de ellos. Para esto, se utilizan cadenas de ceros y unos que corresponderían a la salida característica de los sensores que detectan la presencia o ausencia de un objetivo [84, 85]. Se colocan ceros para los objetos no disponibles y unos para los disponibles, bajo la suposición de que las cadenas de ceros y unos corresponden a un plano rectangular. Es posible que el lugar sea asimétrico, en este caso, se completa el plano con ceros para indicar que existen espacios no disponibles o no existentes y de la misma manera, los obstáculos que se encuentren en el plano o espacios no utilizados se tomarán como ceros.

Para los datos espaciales, se tiene el archivo **aPtosRef** que contendrá las coordenadas iniciales, relativas al plano, de los puntos de referencia, suponiendo que el punto (0,0) está en la esquina superior izquierda del plano. Se coloca al final del archivo, el identificador que corresponde al punto de referencia móvil para distinguirlo.

El archivo **aDimNoEsp** mantiene los valores de las dimensiones no espaciales, los cuales no cambian en el tiempo. Estos valores deben ser distintos entre sí para que la propiedad $S(A) \subset S(A, Q)$ [1] se mantenga, la cual permite que los algoritmos **VC2S+** y **CD2S** puedan calcular aparte el *skyline* espacial $S(A)$. Si se tuvieran dos objetos iguales en todas las dimensiones no espaciales, por ejemplo x y y , y estos forman parte del *skyline* no espacial

$x, y \in S(A)$, puede ocurrir que x domine a y tomando en cuenta las dimensiones espaciales (las distancias a los puntos de referencia) y en ese caso sólo x estaría en el *skyline* final, devuelto por el algoritmo que combina el *skyline* espacial y no espacial $x \in S(A, Q)$, pero y dejaría de formar parte del *skyline* $y \notin S(A, Q)$. Entonces, la propiedad antes mencionada no podría utilizarse.

Para mayor información acerca de los archivos de entrada, referirse al Apéndice A donde se encuentra un ejemplo de estos archivos.

3.1.1 Simulador

Debido a que no se dispone de sensores, ni de GPS, en este trabajo de grado se realizan los cambios que pueden ocurrir en el ambiente mediante un simulador, el cual trabaja en base a abstracciones de estos posibles ambientes reales. Se entiende por simulador, al componente capaz de transformar un ambiente ficticio de entrada en otro, aplicando uno de los siguientes cambios:

- * **Cambios en la disponibilidad:** se escoge aleatoriamente, según el porcentaje de cambio dado, `%Cambio`, los objetos candidatos a ser modificadas, asignando el estatus disponible a los objetos ocupados y el estatus ocupado a los disponibles.
- * **Desplazamiento del punto de referencia:** la posición del punto de referencia móvil puede variar en línea recta escogiendo de forma aleatoria hacia arriba, abajo, izquierda o derecha según la distancia máxima dada `DistMax` y las dimensiones del plano de objetos, evitando así que el punto de referencia móvil quede fuera de los límites del plano o estancado en alguna esquina.

Por consiguiente, los algoritmos implementados en este trabajo de grado, emplean tres modos de ejecución para encontrar el *skyline* según sea el caso de simulación. Se entiende como simulación a la salida del Simulador para el ambiente asignado como entrada. Para esto se invocan los siguientes métodos que se sobrescriben con la implementación de cada algoritmo:

- * **comenzar()**: obtiene el conjunto *skyline* espacial y no espacial para la configuración inicial de entrada (*modo inicial*).
- * **calcularSolucionCambioEstatus()**: obtiene el conjunto *skyline* espacial y no espacial para el ambiente resultante luego de una transformación del conjunto de objetos disponibles y no disponibles (*modo actualización de estatus*).
- * **calcularSolucionDesplazamiento()**: obtiene el conjunto *skyline* espacial y no espacial para el ambiente resultante luego de un cambio de posición del punto de referencia móvil (*modo actualización de desplazamiento*).

El Simulador se encarga entonces de alterar el ambiente según los datos iniciales `aConf`, `aPtosRef`, `aDimNoEsp` y los parámetros dados. Tales parámetros son: la cantidad de simulaciones `n`, los modos de simulación permitidos `modo`, el cual puede ser combinado, sólo *modo actualización de estatus* o sólo *modo actualización de desplazamiento*, el porcentaje de cambio de disponibilidad `%Cambio` y la máxima distancia que se puede desplazar el punto móvil en una simulación `DistMax`. Para mayor detalle, ver el Apéndice B.

3.1.2 Ejemplo de posibles simulaciones

Con el propósito de explicar intuitivamente las tareas que realizan los algoritmos implementados, se presenta un ejemplo de tres simulaciones para el caso de cero dimensiones no espaciales (Ejemplo I) y otro para dos dimensiones no espaciales (Ejemplo II).

Ambos ejemplos contienen 400 puestos de un estacionamiento ocupados en un 20 % inicialmente, tres puntos de referencia estáticos o las salidas peatonales del estacionamiento y un punto de referencia móvil o posición inicial del vehículo.

3.1.2.1 Ejemplo I: Dimensiones sólo espaciales

La Figura 3.1 representa una foto del estacionamiento con los datos iniciales una vez encontrados los puestos interesantes como resultado de la ejecución del Simulador (Apéndice B, Algoritmo 13, Líneas 3-9). Los puestos disponibles se colocan como puntos verdes y los ocupados como puntos rojos; los puntos de referencia son los círculos medianos amarillos,

el punto de referencia móvil está resaltado en un recuadro naranja y, los círculos resaltados en verde forman el *skyline*, el cual debe ser la salida de cada uno de los algoritmos al terminar su ejecución. El resultado del *skyline* es el conjunto de 14 puestos interesantes por estar cercanos a los puntos de referencia.



Figura 3.1: Ejemplo I, *modo inicial* (simulación 0)

Supóngase ahora que ocurre un cambio de disponibilidad en los puestos: 217 pasan a estar ocupados y 52 a estar disponibles. Los algoritmos (Líneas 11-16), deben conseguir el nuevo conjunto de puestos interesantes, el cual disminuye a 8 puestos como se muestra en la Figura 3.2. Posteriormente, imagínese que el punto móvil se desplazó en línea recta



Figura 3.2: Ejemplo I, *modo actualización de estatus* (simulación 1)

hacia abajo del punto (4,17) al punto (9,17). En este caso, el conjunto de candidatos disponibles se mantiene, pero los algoritmos (Líneas 17-23), deben conseguir el nuevo *skyline* resultante del desplazamiento del punto de referencia, el cual aumenta ahora a 17 puestos interesantes como se presenta en la Figura 3.3.



Figura 3.3: Ejemplo I, *modo actualización de desplazamiento* (simulación 2)

3.1.2.2 Ejemplo II: Dimensiones espaciales y no espaciales

Considere que ahora se desean obtener los mejores puestos cercanos a las salidas peatonales del estacionamiento y al vehículo, pero que además tengan un bajo tiempo de salida con el vehículo y un bajo precio por hora, los algoritmos (Líneas 3-9) deben conseguir los mejores puestos. El resultado del *skyline* es el conjunto de 70 puestos interesantes por estar cercanos a los puntos de referencia y además tener los mejores tiempo de salida y precios, observable en la Figura 3.4.

Aunque este ejemplo inicialmente tiene la misma cantidad de puestos ocupados que el Ejemplo I (80 puestos), el hecho de que existan dos dimensiones adicionales para calcular el *skyline*, hace que exista una mayor cantidad de puestos interesantes porque se tienen más dimensiones [3].

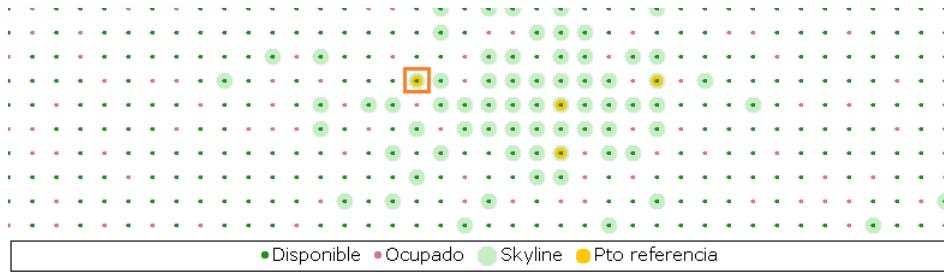


Figura 3.4: Ejemplo II, *modo inicial* (simulación 0)

Si existe un cambio de disponibilidad donde 196 puestos pasan a estar ocupados y 53 a estar disponibles, los algoritmos (Líneas 11-16) obtendrán los 36 puestos interesantes presentados en la Figura 3.5.

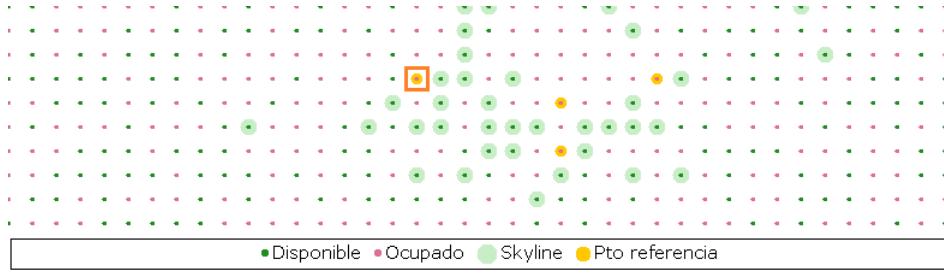


Figura 3.5: Ejemplo II, *modo actualización de estatus* (simulación 1)

Por último, si el punto de referencia se desplazó, del punto (3,17) al (3,20) en línea recta hacia la derecha, los algoritmos (Líneas 17-23) deben retornar los 27 puestos interesantes que aparece en la Figura 3.6.



Figura 3.6: Ejemplo II, modo actualización de desplazamiento (simulación 2)

3.1.3 Preliminares

A continuación, en las secciones 3.1.3.1, 3.1.3.2 y 3.1.3.3 se presentan las bases de los algoritmos implementados.

3.1.3.1 BNL

El algoritmo *Block Nested Loops* (BNL) [3], es utilizado por los algoritmos 2B2S, IB2Sy CD2S como parte del procedimiento para conseguir el *skyline* al problema DSS.

El BNL es el algoritmo más básico para obtener el *skyline*. El algoritmo dispone de una lista de **candidatos** y la ventana **skyline** en la que coloca los objetos incomparables hasta el momento.

Inicialmente, la ventana **skyline** esta vacía por lo que el algoritmo agregará el primer candidato al **skyline**. En cada iteración, se accede a las dimensiones del candidato en cuestión para la evaluación de dominancia. En esta evaluación, se compara el objeto de la iteración contra cada uno de los objetos de la ventana **skyline**, verificando, por cada dimensión, si el objeto candidato domina, es dominado o es igual en esa dimensión al objeto **skyline** en revisión. Si el objeto **skyline** es dominado, lo elimina de la ventana; si el candidato es dominado, lo descarta y continua con el siguiente candidato; si los objetos son incomparables entre sí, pasa a la comparación con el siguiente objeto de la ventana. Si no quedan más objetos por revisar en la ventana, el candidato es incomparable con todos los pertenecientes a la ventana, entonces el algoritmo lo agrega a la ventana. Una vez verificados todos los candidatos, culmina el algoritmo y devuelve los objetos del **skyline**.

3.1.3.2 SFS

El algoritmo *Sort Filter Skyline* (SFS) [22] es un algoritmo progresivo, el cual realiza un ordenamiento topológico del conjunto de datos sobre las dimensiones del *skyline* para garantizar que cada elemento que ingrese en la ventana del *skyline* no sea dominado posteriormente por los candidatos que quedan por revisar.

En una fase previa, ordena los candidatos en función de las dimensiones. Posteriormente, ejecuta el procedimiento que realiza el algoritmo BNL a excepción de que cuando un objeto es agregado a la ventana, no puede ser eliminado por otro.

En síntesis, se realizan menos operaciones en la ventana ya que no necesita eliminar objetos de allí. Este algoritmo es empleado en CD2S.

3.1.3.3 VS^2 y VCS^2

Existen ciertas propiedades geométricas que pueden ser aplicadas para la minimización de comparaciones de dominancia necesarias para la obtención del *skyline espacial*.

Estas propiedades se centran en el *convex hull*, el cual en este caso, es el polígono convexo formado por el conjunto de puntos de referencia, donde estos puntos pueden ser vértices o estar dentro del mismo. Cada uno de los puntos de referencia forma una región de visibilidad que determina la pertenencia o no al *skyline*. Los puntos de referencia que no son vértices del *convex hull* no influyen en el cálculo del *skyline*, esto se debe a que la región de visibilidad de esos puntos estará contenida en la región de visibilidad formada por los vértices. Más aún, todo punto que esté dentro de este *convex hull* es un punto *skyline* y las celdas Voronoi de cada objeto que tengan intersección con el *convex hull* también son *skyline* [1].

Estas propiedades son la base de los algoritmos propuestos en [1]: Voronoi-Based Spatial Algorithm VS^2 y Voronoi-Based Continuous SSQ VCS^2 . El algoritmo VS^2 mediante la construcción de un diagrama de Voronoi [86], utiliza el grafo *Delaunay* [87] correspondiente

para navegar entre los objetos necesarios y así calcular el *skyline* espacial. Este algoritmo permite únicamente puntos de referencia estáticos, para la manipulación de puntos de referencia dinámicos, los autores utilizan el algoritmo *VCS*² que realiza la actualización continua del *skyline* espacial originado por el desplazamiento de uno de los puntos de referencia a la vez y sólo en algunos casos recurre al algoritmo *VS*².

El algoritmo *VCS*² en una fase previa construye un diagrama de Voronoi y el grafo *Delau-nay* de la configuración inicial. A partir del grafo, obtiene el *skyline* espacial tomando los puntos cuyas celdas de Voronoi se intersecten con el *convex hull* o estén dentro del mismo.

El algoritmo permite el desplazamiento de un punto de referencia a la vez, comparando la posición anterior con la actual al existir una variación e identifica el patrón de cambio y realiza la acción según sea el caso.

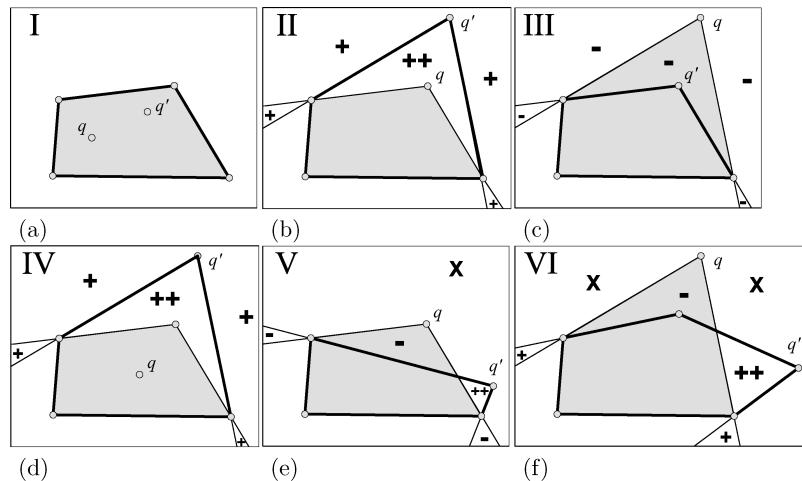


Figura 3.7: Patrones de cambio del *convex hull* de Q cuando q cambia a q' . Fuente [1]

Sean Q el conjunto de puntos de referencia, Q' el nuevo conjunto de posiciones, q el punto en desplazamiento y q' la nueva posición de q :

El patrón I de la Figura 3.7(a), corresponde al caso en que el *convex hull* anterior $CH(Q)$ y el actual $CH(Q')$ son iguales ya que ambas posiciones q y q' del punto móvil están dentro del *convex hull*. En este caso el *skyline* no se modifica, por lo que no es necesario atravesar el grafo.

Para los patrones II-V, se atraviesa el grafo en las regiones indicadas, las regiones en blanco no deben ser examinadas debido a que están fuera de región de visibilidad de q y q' .

En los patrones II de la Figura 3.7(b) y IV Figura 3.7(d), $CH(Q)$ está contenido en $CH(Q')$, por lo que la región $CH(Q)$ será igual a $CH(Q) \cap CH(Q')$ implicando que los objetos en $CH(Q)$ que pertenecían al *skyline* se mantienen porque están dentro de $CH(Q')$. Por lo tanto, es únicamente necesario la agregación al *skyline* de aquellos objetos, marcados con $++$, que estén en $CH(Q') - CH(Q)$ y la examinación de los objetos marcados con $+$ para verificar su dominancia y decidir si deben ser agregados al *skyline*, ya que debido al desplazamiento de q' la región de dominancia cambió y son más los candidatos que pueden pertenecer al *skyline*.

En el patrón III de la Figura 3.7(c), $CH(Q)$ contiene a $CH(Q')$, por lo que la región $CH(Q')$ será igual a $CH(Q) \cap CH(Q')$, lo que significa que se mantienen los objetos de la intersección. Los objetos que estén fuera de $CH(Q')$ marcados con $-$ deben ser examinados para verificar su dominancia y decidir si deben ser eliminados del *skyline*.

En el patrón V de la Figura 3.7(e), prevalecen en el *skyline* los objetos pertenecientes a $CH(Q) \cap CH(Q')$. Deben agregarse los que estén dentro de $CH(Q') - CH(Q)$ y debido a que la región de dominancia cambió, es necesario verificar si deben ser eliminados los objetos fuera del $CH(Q')$ marcados con $-$ y agregados o eliminados los marcados con x .

Para los demás casos, como el patrón VI de la Figura 3.7(f), no se actualiza el conjunto *skyline* según el cambio de posición del punto de referencia móvil, sino que se ejecuta desde el inicio el algoritmo VS^2 que toma todos los puntos de referencia como estáticos, por lo que no realiza optimizaciones en el cálculo del *skyline* en relación al desplazamiento que ocurrió.

Este algoritmo ha sido adaptado para el problema DSS, denominado **VC2S+**. Así como también el método de reconocimiento del patrón de cambio es empleado en el algoritmo **CD2S**.

3.2 Soluciones basadas en BNL

Este algoritmo fue implementado como primera solución al problema DSS (Algoritmo 1-3). Esta solución se denomina *Basic BNL for DSS* (2B2S, por sus siglas en inglés) debido a que utiliza el algoritmo BNL [3] para el cálculo del *skyline*, pero no dispone de métodos para actualizar el conjunto *skyline* según cambios de disponibilidad o posición del punto de referencia móvil, por lo que debe reiniciar la construcción del *skyline* en cada modo de ejecución para obtener los objetos correctos. El algoritmo explora de forma secuencial el plano en su totalidad para encontrar los candidatos al *skyline*.

3.2.1 Ejemplo de ejecución del algoritmo 2B2S

Utilizando las simulaciones descritas en la sección 3.1.2, se explica intuitivamente el algoritmo 2B2S.

3.2.1.1 Ejemplo I: Dimensiones sólo espaciales

La primera ejecución del algoritmo 2B2S se realiza en el *modo inicial*, en la cual se revisa el plano completo en busca de los puestos disponibles. A partir de los puestos disponibles, 2B2S calcula sus distancias correspondientes a los 4 puntos de referencia y con ellos genera el *skyline*. El *skyline* resultante se observa en la Figura 3.1, p. 43.

Con el cambio de disponibilidad, el algoritmo se 2B2S ejecuta en el *modo actualización de estatus* para actualizar el estatus de cada uno de los puestos en el plano, por lo tanto cambia el conjunto de candidatos disponibles y con ellos, obtiene el *skyline* mostrado en la Figura 3.2, p. 43.

A causa del desplazamiento del punto de referencia móvil, es necesario ejecutar el algoritmo en el *modo actualización de desplazamiento* para obtener el resultado correcto. El algoritmo actualiza la nueva distancia con respecto al punto de referencia que se desplazó de cada uno de los candidatos. Una vez corregida la distancia, se genera el *skyline* presentado en la Figura 3.3, p. 43.

3.2.1.2 Ejemplo II: Dimensiones espaciales y no espaciales

Tomando la existencia de dimensiones no espaciales, la primera ejecución del algoritmo se realiza de forma similar al Ejemplo I, pero incluye la lectura y almacenamiento de estas dimensiones no espaciales en los atributos del objeto. Estos datos, son utilizados al momento de realizar las comparaciones de dominancia con las dimensiones espaciales y no espaciales para obtener el *skyline* de la Figura 3.4, p. 44.

Para el cambio de disponibilidad y el desplazamiento del punto de referencia mostrados en las Figuras 3.5 y 3.6, p. 44, se realiza exactamente el mismo proceso que en el Ejemplo I para el *modo actualización de estatus* y *modo actualización de desplazamiento*, pero considerando también las no espaciales en la verificación de dominancia.

3.2.2 Estructuras utilizadas por el algoritmo 2B2S

El plano correspondiente a la disposición de los objetos es representado con una matriz de *bytes*, **plano**, cuyas dimensiones se calculan tomando los límites del plano de entrada: el largo corresponde a la cantidad de filas y el ancho dividido entre el tamaño de un *byte* determina la cantidad de columnas. Cada *byte* del archivo de entrada **aConf** es almacenado en una casilla de la matriz, por lo que una casilla en la matriz contendrá el estatus de ocho objetos contiguos. La matriz en la Figura 3.1 corresponde a los puntos pequeños verdes y rojos, donde el color indica el estatus que posee.

En cuanto a los puntos de referencia del *skyline* espacial definidos en **aPtosRef**, se utiliza una lista de puntos, **ptosRef**. Estos puntos son ingresados en el *modo inicial* y, en el *modo actualización de desplazamiento* se actualiza sólo el registro correspondiente al índice **iPtoDin**, el cual indica la posición en la lista del punto de referencia móvil. Estos puntos en las Figura 3.1 corresponden a los círculos amarillos.

Adicionalmente, los atributos de cada objeto son almacenados en la lista **attObj**, utilizando como índice el número de la casilla del objeto en el plano. Este número se calcula a partir de las coordenadas *x* y *y* del centro del objeto y las dimensiones del plano.

Estos atributos comprenden:

- * **Coordenadas:** posición (x,y) del centro del objeto, la cual es relativa al plano.
- * **Dimensiones espaciales:** son los valores resultantes de calcular la distancia euclídea del centro del objeto a cada punto de referencia. Estos datos son almacenados la primera vez que se explora el objeto y se actualiza sólo la dimensión correspondiente al punto de referencia en desplazamiento, para evitar el procesamiento innecesario de estas distancias en las ejecuciones posteriores.
- * **Dimensiones no espaciales:** son los valores definidos en el archivo de dimensiones no espaciales de entrada `aDimNoEsp`. Estas dimensiones, se obtienen una sola vez en el *modo inicial* y no son modificados. El número de dimensiones no espaciales es colocado en la constante `dNoEsp`.
- * **Etiquetas:** codificación en *bits* para identificar la pertenencia al `skyline` y a la lista de `candidatos` y así evitar la búsqueda del objeto en las estructuras.

Por otra parte, se mantiene una lista de `candidatos` que almacena los números de las casillas que identifiquen los objetos disponibles en el plano. Estos corresponden en la Figura 3.1 a las casillas de los puntos pequeños verdes.

Por último, respecto a los objetos que conformarán el *skyline*, las referencias a los elementos en `attObj` de aquellos objetos considerados incomparables serán almacenadas en la lista `skyline`. Estos corresponden en la Figura 3.1 a los círculos verdes.

3.2.3 Descripción del algoritmo 2B2S

A continuación se describe el algoritmo 2B2S por modo de ejecución.

3.2.3.1 Modo inicial

El algoritmo 1 inicia su ejecución en *modo inicial* con el procedimiento COMENZAR (Líneas 4-7) el cual procesa los archivos indicados en los parámetros de entrada: `aConf`, `aPtosRef`

y `aDimNoEsp` para inicializar las estructuras, luego busca los objetos disponibles y con estos calcula el *skyline*. Para esto invoca los métodos: `INICIALIZAR` y `CALCULARSOLUCION`.

El procedimiento `INICIALIZAR` (Líneas 8-13) procesa los archivos de entrada y asigna los valores iniciales a las variables y estructuras. Este procedimiento, llena la lista `ptosRef` con las coordenadas de los puntos de referencia del *skyline* espacial y el apuntador al punto de referencia móvil en `iPtoDin` (Línea 10), construye la matriz `plano` con los estatus iniciales (Línea 11).

En caso de existir dimensiones no espaciales (Línea 12), el procedimiento `INICIALIZAR` asigna los valores dados de estas dimensiones para cada objeto del plano en la lista `attObj` y, coloca el número de dimensiones en la variable `dNoEsp`. Si no existen dimensiones no espaciales, simplemente inicializa esta lista.

Posteriormente, en `CALCULARSOLUCION` (Líneas 14-20), se recorre la matriz `plano` en forma secuencial, *byte* a *byte*, identificando los objetos disponibles.

Si el *byte* `plano[i][j]` es igual a 0, entonces continua con la examinación del siguiente *byte*; si consigue alguno distinto de 0, revisa cuáles *bits* del *byte* están encendidos y por cada *bit* en 1 crea una entrada en la lista `candidatos` (Línea 16).

En el *modo inicial*, como aún no se conocen las dimensiones espaciales del objeto en `attObj`, el algoritmo calcula las distancias del objeto a cada uno de los puntos de referencia y las almacena allí; en el resto de los modos de ejecución, en caso de que estas distancias se hayan calculado previamente, sólo se actualiza la correspondiente al punto de referencia en desplazamiento.

Una vez finalizada la exploración de candidatos, se realizan las comparaciones del *skyline* con el algoritmo `BNL`, iterando la lista `candidatos` y accediendo a los atributos espaciales y no espaciales en `attObj` del candidato, mediante `OBTENERSKYLINE` (Línea 18).

Algoritmo 1 2B2S (Parte 1)

```

1: Integer iPtoDin, dNoEsp
2: Byte[ ][ ] plano
3: List<Object> candidatos, skyline, attObjs, ptosRef
4: procedure COMENZAR(aConf, aPtosRef, aDimNoEsp)           ▷ Calcula el primer Skyline (Modo inicial)
5:   INICIALIZAR(aConf, aPtosRef, aDimNoEsp)
6:   return CALCULARSOLUCION(skyline, candidatos, plano)
7: end procedure

8: procedure INICIALIZAR(aConf, aPtosRef, aDimNoEsp)          ▷ Lee los archivos de entrada
9:   candidatos, skyline  $\leftarrow \emptyset$ 
10:  ptosRef  $\leftarrow$  OBTENERPUNTOSREFERENCIA(aPtosRef, iPtoDin)
11:  plano  $\leftarrow$  OBTENERDISPONIBILIDADOBJETOS(aConf)
12:  attObjs  $\leftarrow$  OBTENERVALORESDIMENSIONESNOESPAZIALES(aDimNoEsp, dNoEsp)
13: end procedure

14: procedure CALCULARSOLUCION(skyline, candidatos, plano)      ▷ Busca candidatos, obtiene el skyline
15:   for byte  $\in$  plano do          ▷ Recorre byte a byte, obtiene los disponibles, los agrega y calcula sus distancias
16:     AGREGARCANDIDATOS(candidatos, byte)
17:   end for
18:   skyline  $\leftarrow$  OBTENERSKYLINE(candidatos)                         ▷ Utiliza el algoritmo Skyline BNL
19:   return skyline
20: end procedure

```

3.2.3.2 Modo actualización de estatus

En el *modo actualización de estatus* el algoritmo 2, inicia la ejecución con el procedimiento CALCULARSOLUCIONCAMBIOESTATUS (Líneas 21-25) que reinicia las estructuras *skyline* y *candidatos*. Con el procedimiento ACTUALIZARSTATUS se sincroniza el *plano* con los nuevos estatus de disponibilidad de los objetos indicados en este modo de ejecución y nuevamente se procesa los datos mediante CALCULARSOLUCION para conseguir el *skyline*.

Algoritmo 2 2B2S (Parte 2)

```

21: procedure CALCULARSOLUCIONCAMBIOESTATUS(ocupados, disponibles)    ▷ Modo actualización de estatus
22:   candidatos, skyline  $\leftarrow \emptyset$ 
23:   ACTUALIZARSTATUS(plano, ocupados, disponibles)                  ▷ Actualiza la disponibilidad en el plano
24:   return CALCULARSOLUCION(skyline, candidatos, plano)
25: end procedure

```

3.2.3.3 Modo actualización de desplazamiento

Finalmente, en caso del *modo actualización de desplazamiento* el algoritmo 3, inicia la ejecución con el procedimiento CALCULARSOLUCIONDESPLAZAMIENTO (Líneas 26-30) que

reinicia la ventana *skyline*, pero mantiene la lista *candidatos*, la itera y a cada uno de sus elementos les actualiza la distancia al punto de referencia móvil invocando a ACTUALIZARDISTANCIAPTODIN. Por último, este procedimiento consigue el *skyline* con CALCULARSOLUCION.

Algoritmo 3 2B2S (Parte 3)

```

26: procedure CALCULARSOLUCIONDESPLAZAMIENTO(nuevaPosPtoRef)           ▷ Modo actualización de desplazamiento
27:   candidatos, skyline  $\leftarrow \emptyset$ 
28:   ACTUALIZARDISTANCIAPTODIN(attObjs, nuevaPosPtoRef)      ▷ Cambia solo esa dist en candidatos
29:   return CALCULARSOLUCION(skyline, candidatos, plano)
30: end procedure

```

3.2.4 Algoritmo IB2S

Debido a que el algoritmo 2B2S no actualiza el conjunto *skyline* según los cambios ocurridos en el ambiente eliminando o agregando objetos al *skyline*, debe calcularlo desde el inicio en todos los modos de ejecución y esto incurre en un alto número de comparaciones de dominancia. Es así que, considerando una mejora al algoritmo al menos en el *modo actualización de estatus*, se propuso el Algoritmo 4.

El algoritmo 4, se denomina *Improved BNL for DSS* (IB2S, por sus siglas en inglés) ya que similar al algoritmo 2B2S utiliza la técnica BNL para el cálculo del *skyline* con la diferencia que en el *modo actualización de estatus* en lugar de reiniciar el conjunto *skyline*, lo actualiza según las nuevas condiciones. El algoritmo IB2S utiliza las mismas estructuras y procedimientos que el algoritmo 2B2S a excepción del procedimiento calcularSolucionCambioStatus que es sobreescrito y la adición de los nuevos procedimientos llamados en él.

3.2.4.1 Modo actualización de estatus

Inicialmente, en el procedimiento invocado por el *modo actualización de estatus* (Líneas 1-29) se sincroniza el *plano* con los nuevos estatus de los objetos (Línea 3). Seguidamente, el algoritmo itera sobre la lista *ocupados* para descartar a través de *eliminarCandidato* y de *eliminarObjSkyline* de la lista de *candidatos* y del *skyline* los objetos que ya no estarán disponibles, los cuales serían los nuevos puestos ocupados (Líneas 4-10).

Si no se eliminan objetos del *skyline* (Líneas 12-15), en los **candidatos** existentes (sin incluir los nuevos puestos disponibles), no habrá uno mejor o incomparable a los que ya están en el *skyline*, de lo contrario pertenecerían al *skyline* del que partió en este modo de ejecución. Al no existir cambios en el *skyline*, el conjunto de dominados se mantiene, por lo que no será necesario revisar los **candidatos**. La lista **candidatos** se almacena temporalmente en **candidatosTmp** y se reinicia para prepararse a los nuevos elementos en **disponibles**.

De lo contrario, si se eliminan objetos del *skyline* debido al cambio de disponibilidad ocurrido, podrían existir algunos candidatos, dominados anteriormente (por los objetos que pasaron a ocupados), que tomen ahora ese lugar. En este caso, se mantiene la lista **candidatos** para revisarla y conseguir esos objetos.

Posteriormente, a través de **agregarCandidato** (Líneas 16-19), el algoritmo examina los objetos en la lista **disponibles**, que serían los nuevos puestos disponibles y los agrega a la lista **candidatos**. Para cada objeto que antes no haya estado disponible, se calculan todas las distancias correspondientes a las dimensiones espaciales y para los objetos que ya estuvieron disponibles alguna vez, sólo se actualiza la distancia al punto en desplazamiento. Este proceso se realiza porque el punto de referencia móvil pudo haber sufrido alguna alteración en su posición desde que el objeto dejó de estar disponible.

Si no existen nuevos candidatos y no se eliminaron objetos del *skyline* (Líneas 20-21), entonces copia la lista **candidatosTmp** a **candidatos** y finaliza devolviendo la lista **skyline** original ya que el *skyline* no fue alterado.

En caso de existir nuevos candidatos o elementos eliminados del *skyline* porque cambiaron a ocupados (Líneas 22-27), se procede a actualizar el *skyline* mediante **obtenerSkyline**, cuya función conservando los objetos que están en **skyline**, eliminará objetos del mismo, descartará o agregará nuevos al conjunto según sea el resultado de las comparaciones de dominancia utilizando la lista **candidatos**. Finalmente, devuelve el *skyline* actualizado.

Algoritmo 4 IB2S

```

1: procedure CALCULARSOLUCIONCAMBIOESTATUS(ocupados, disponibles)           ▷ Modo actualización de estatus
2:   skyline  $\leftarrow \emptyset$ 
3:   ACTUALIZARESTATUS(plano, ocupados, disponibles)                                ▷ Actualiza la disponibilidad en el plano
4:   Integer elimSkyline  $\leftarrow 0$                                                  ▷ Contador de eliminados
5:   for o  $\in$  ocupados do          ▷ Elimina los nuevos objetos ocupados de las listas skyline y candidatos
6:     if o  $\in$  skyline then
7:       ELIMINAROBJSKYLINE(skyline, o, elimSkyline)
8:     end if
9:     ELIMINARCANDIDATO(candidatos, o)
10:    end for
11:    List <Object> candidatosTmp  $\leftarrow \emptyset$ 
12:    if elimSkyline = 0 then ▷ Si no se eliminaron objetos del skyline, solo hay que revisar los nuevos candidatos
13:      candidatosTmp  $\leftarrow$  candidatos
14:      candidatos  $\leftarrow \emptyset$ 
15:    end if
16:    Integer agrCandidatos  $\leftarrow 0$                                               ▷ Contador de agregados
17:    for d  $\in$  disponibles do          ▷ Agrega los nuevos objetos disponibles
18:      AGREGARCANDIDATO(candidatos, d, agrCandidatos)
19:    end for
20:    if elimSkyline = 0 and agrCandidatos = 0 then ▷ Si no se eliminaron, ni agregaron objetos del skyline
21:      candidatos  $\leftarrow$  candidatosTmp                                              ▷ Se mantiene el skyline
22:    else
23:      skyline  $\leftarrow$  OBTENERSKYLINE(candidatos)                                     ▷ Se modifica el skyline
24:      if elimSkyline = 0 then
25:        ADDALL(candidatos, candidatosTmp)                                         ▷ Agrega los nuevos a candidatos
26:      end if
27:    end if
28:    return skyline
29: end procedure

```

3.3 Algoritmo VC2S+

Debido a que los algoritmos *VS²* y *VCS²* no contemplan la presencia de dimensiones no espaciales en los datos y no consideran un cambio de estatus en los objetos de entrada o al menos la posibilidad de agregar o eliminar estos objetos de entrada, se implementaron estos algoritmos con una adaptación que admite estas preferencias sobre múltiples atributos del objeto y los cambios de disponibilidad del mismo. Este algoritmo se denominó Voronoi-Based Continuous SSQ Plus (**VC2S+**).

La adaptación al problema DSS (Algoritmos 5-9), consiste para las dimensiones no espaciales en los cambios propuestos por los autores en [1], estos son: calcular aparte el *skyline* de estas dimensiones, *skyline* no espacial, incluir en la verificación de dominancia estas dimensiones y ampliar la región de búsqueda para examinar los candidatos al *skyline*.

Adicionalmente, para los cambios de disponibilidad, se propuso en este trabajo de grado, considerar sólo los objetos disponibles en la creación del diagrama de Voronoi, ya que si se agregan los objetos no disponibles alterarían el *skyline* produciendo resultados incorrectos. La principal desventaja de esta adaptación es que cada vez que se ejecute el *modo actualización de estatus*, se tendrá que construir el diagrama de Voronoi que refleje estos cambios, recrear el grafo *Delaunay* asociado y además aplicar el algoritmo *VS*² adaptado desde el comienzo.

3.3.1 Ejemplo de ejecución del algoritmo VC2S+

Utilizando las simulaciones en la sección 3.1.2, se explica el algoritmo VC2S+.

3.3.1.1 Ejemplo I: Dimensiones sólo espaciales

En el *modo inicial*, se obtienen los puestos disponibles con los que se construye el grafo *Delaunay*, el cual se muestra en la Figura 3.8, con líneas de color gris.

No se observan los puestos ocupados debido a que el grafo únicamente tiene aquellos objetos que el algoritmo podría considerar para el *skyline*, si se dejaran los puestos ocupados, la forma en que se revisa el grafo y se poda la búsqueda no funcionaría correctamente y arrojaría resultados incorrectos.

El *convex hull* que es el polígono negro que encierra a los puntos de referencia (círculos amarillos) es construido y para conseguir el *skyline*, se ejecuta la versión implementada del algoritmo *VS*².

El algoritmo inicia la revisión del grafo con el punto más cercano al punto de referencia móvil, lo coloca como visitado (región en gris de la figura), si el punto está dentro del *convex hull* lo agrega al *skyline* y si no pertenece, verifica si el objeto no es dominado por el resto de los objetos visitados que son incomparables entre sí, que hasta el momento forman un posible el *skyline*, para agregarlo a esa pila. Posteriormente, evalúa sus vecinos, en caso de encontrar alguno que sea dominado, lo descarta.

Cuando no hayan más objetos a ser explorados, se suspende la búsqueda. Una última verificación es realizada entre los posibles objetos *skyline* y los que pertenecen al *convex hull* para descartar dominados. Finalmente el algoritmo VC2S+, retorna los mismos 14 puestos interesantes, de los cuales cinco pertenecen al *convex hull* (SkylineDin).

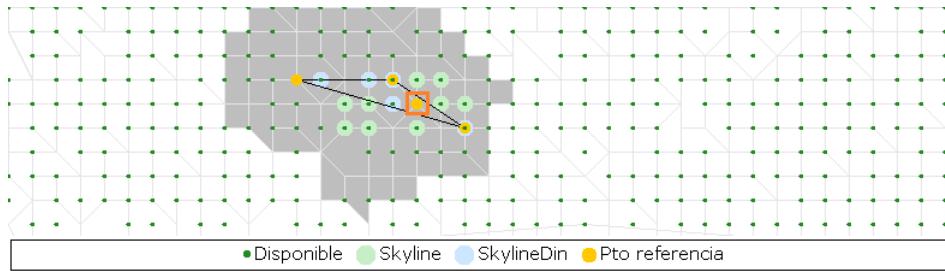


Figura 3.8: Ejemplo I, *modo inicial* (simulación 0)

Por otra parte, el algoritmo en el *modo actualización de estatus*, elimina los puestos disponibles que pasaron a estar ocupados, con ellos vuelve a crear el grafo *Delaunay* y ejecuta el procedimiento anterior. Resultando cuatro de los ocho puestos *skyline*, pertenecientes al *convex hull*, visibles en la Figura 3.9.

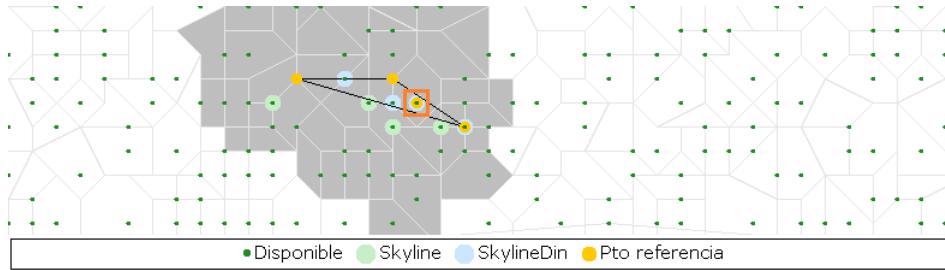
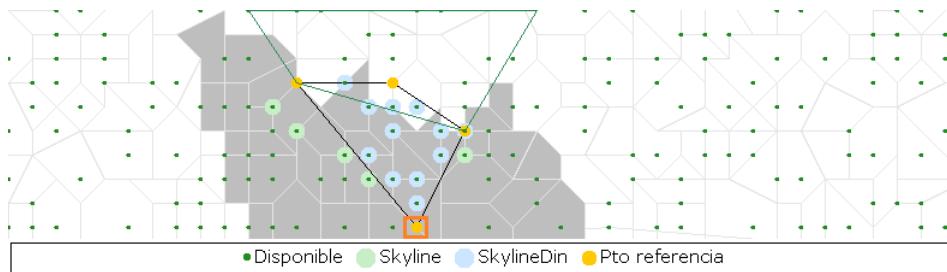


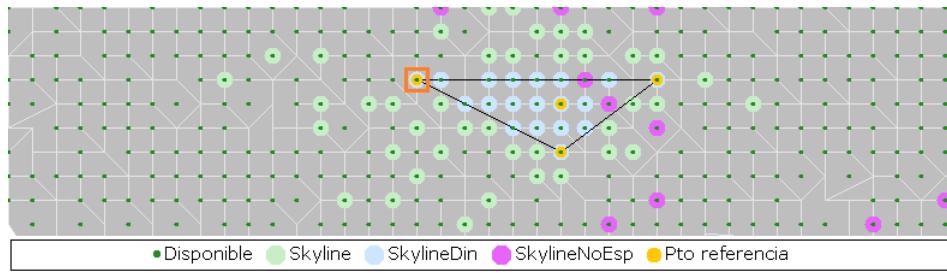
Figura 3.9: Ejemplo I, *modo actualización de estatus* (simulación 1)

En presencia de desplazamiento, el algoritmo en el *modo actualización de desplazamiento*, identifica el patrón de desplazamiento, en este caso es el patrón IV, que mantiene los objetos que ya existían en el *skyline* y sólo es necesario agregar nuevos objetos con el algoritmo *VCS*². No se verifica dominancia para aquellos que están ahora en el nuevo *convex hull*. Los objetos que están fuera del *convex hull* deben ser verificados, pero sólo algunos, como se observó en la Figura 3.7, p. 47. En este caso, son los objetos que están fuera de la región indicada con el polígono verde en la Figura 3.10. Finalmente, se retorna el *skyline* compuesto por 12 puestos pertenecientes al *convex hull* y cinco fuera del *convex hull*.

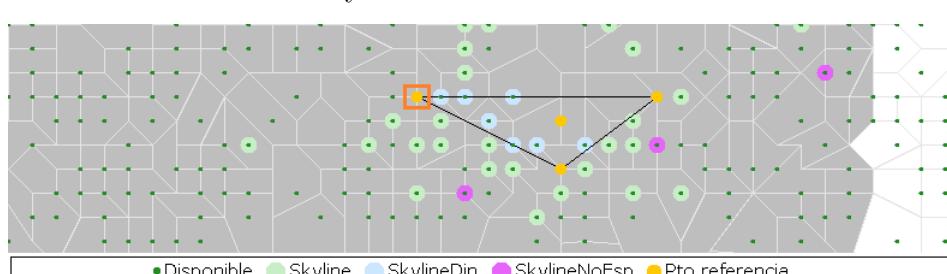


3.3.1.2 Ejemplo II: Dimensiones espaciales y no espaciales

En presencia de dimensiones no espaciales, en el *modo inicial* Figura 3.11, el algoritmo calcula el *skyline* no espacial marcado con círculos magenta considerando sólo las dimensiones no espaciales, retornando 10 de los 70 puestos marcados como interesantes. Con los puestos obtenidos se genera una región de búsqueda que indicará los puestos que deben ser visitados por el algoritmo VS^2 implementado, para conseguir el *skyline*.



Para el *modo actualización de estatus*, se genera el nuevo grafo *Delaunay* y se ejecuta el procedimiento anterior, el cual retorna en la Figura 3.12: 3 puestos del *skyline* no espacial, siete pertenecientes al *convex hull* y 43 fuera del *convex hull*.



En la simulación de desplazamiento, se mantiene el *skyline* no espacial ya que este conjunto sólo cambia si se agregan o eliminan puestos disponibles. Se identifica el patrón de cambio, en este caso es el III, que indica que no se necesita agregar objetos al *skyline*, sólo revisar

los existentes para eliminarlos, pero el algoritmo VCS^2 únicamente revisará la región fuera del polígono verde y del *convex hull* y dentro de la región de búsqueda generada en la simulación anterior. Este proceso retorna en la Figura 3.13, los mismos 3 puestos del *skyline* no espacial, 3 pertenecientes al *convex hull* y 21 fuera del *convex hull*.

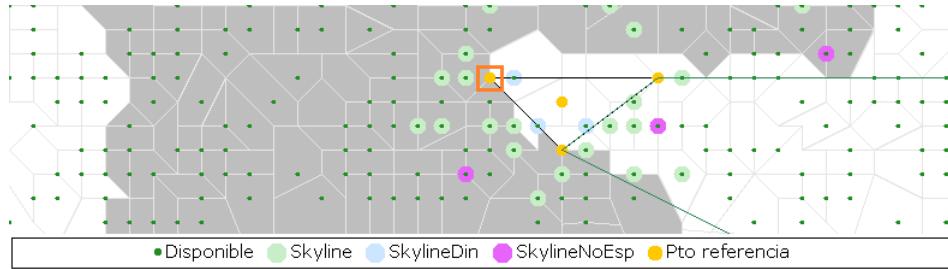


Figura 3.13: Ejemplo II, *modo actualización de desplazamiento* (simulación 2)

3.3.2 Estructuras utilizadas por el algoritmo VC2S+

La estructura **grafo** es destinada para el grafo *Delaunay* de sólo los objetos disponibles. Para esto, es necesario mantener al igual que el algoritmo 2B2S la lista **ATTOBJS** con la información de los atributos de cada objeto (incluyendo los que no están disponibles) y la lista **candidatos** que será utilizada en el momento que sea necesario generar nuevamente el grafo. En la Figura 3.8 el grafo se representa con las líneas grises y los candidatos son las casillas de los puntos pequeños verdes.

Similarmente, se tienen las listas para almacenar los puntos de referencia **ptosRef** y los objetos incomparables en **skyline**. Adicionalmente, se mantiene una lista con el *skyline* no espacial, **skylineNoEsp**, ya que como se indica en [1] se calcula de forma separada para no modificar los algoritmos VS^2 y VCS^2 . En la Figura 3.11 los objetos con círculos magenta corresponden al *skyline* no espacial.

El *convex hull* formado por los puntos de referencia se almacena en la variable **ch** del tipo polígono que se utilizará para identificar el *skyline* espacial. En la Figura 3.8 el polígono negro es el *convex hull*.

En los casos donde existen dimensiones no espaciales, se utiliza además la variable **regionBusqueda** del tipo rectángulo para podar la examinación de los objetos en el grafo.

Con la intención de reducir el procesamiento de datos, se utilizan las listas `celdasVoronoi`, `distanciasVoronoi` y `vecinosVoronoi` que permiten tener calculados e indexados elementos de uso continuo como: la celda Voronoi de un objeto, los valores de las distancias mínimas de la celda a los puntos de referencia y los vecinos Voronoi de ese objeto respectivamente. Sólo si no se encontraren los datos del objeto, serán generados con los procedimientos correspondientes y almacenados en estas estructuras.

3.3.3 Descripción del algoritmo VC2S+

A continuación se describe el algoritmo VC2S+ por modo de ejecución.

3.3.3.1 Modo inicial

El Algoritmo 5, p. 63, de forma similar a los otros algoritmos implementados, se inicia con el procedimiento COMENZAR (Líneas 5-16). Sin embargo, en el procedimiento INICIALIZAR (Líneas 17-22), en lugar de crear la matriz `plano`, agrega a la lista `candidatos` aquellos objetos que están disponibles (Línea 20). Con los disponibles, construye el grafo *Delaunay* (Línea 8). Posteriormente, con los puntos de referencia genera el *convex hull* `ch` (Línea 9).

En presencia de dimensiones no espaciales (Línea 10-12), se procede a calcular el *skyline* no espacial `skylineNoEsp` tomando los objetos candidatos y, genera la `regionBusqueda`. Para mayor información sobre la región de búsqueda, ver el Apéndice C.

Posteriormente, se calcula el *skyline* espacial mediante GENERARVS2 (Línea 13). Este procedimiento corresponde a la implementación del *VS²* (Algoritmo 6, p. 64, Líneas 23-58), que parte, a través de la función OBTENERNN, con la obtención del objeto más cercano al punto de referencia móvil. Agrega el objeto en la pila `h`, ordenada de menor a mayor por el valor `adist` (suma de las dimensiones espaciales) y coloca al objeto en `visitado` para no volverlo a explorar.

Utilizando la pila `h`, se inicia la exploración del grafo *Delaunay* partiendo del objeto encontrado en el paso anterior (Líneas 28-53). Se extrae el último objeto agregado y,

se chequea si está en `skylineNoEsp` (Línea 30). En este paso, se verifica si el objeto está marcado como *skyline* no espacial para evitar una doble revisión, de ser afirmativo, pasaría directamente a la evaluación de sus vecinos Voronoi (Líneas 34-52).

Si el objeto no está en `skylineNoEsp` (Líneas 30-33), se identifica si el objeto está etiquetado como perteneciente al *convex hull* (Línea 31), en caso de ser positivo, el algoritmo lo agrega directamente al `skyline`.

Si el objeto no pertenece al polígono `ch`, a través de la función `ESDOMINADO` (Línea 32), realiza las evaluaciones de dominancia *skyline*, verificando tanto las dimensiones espaciales como las no espaciales, contra los objetos en `skyline` y la pila de probables candidatos al *skyline* `hS`. Si consigue que el objeto es dominado, suspende la verificación y pasa a la evaluación de sus vecinos Voronoi (Líneas 34-52). Sino, lo agrega a la pila `hS`.

Una vez visitado y verificado el objeto, se buscan sus vecinos Voronoi para evaluarlos posteriormente (Línea 34). Para cada vecino, se revisa si ya fue explorado (Línea 36). Si fue visitado, se continúa con el siguiente vecino, sino, se marca como visitado. Si se tienen dimensiones no espaciales (Líneas 38-50), se verifica si el objeto está dentro de la región de búsqueda, en caso de no estarlo, se descarta y se pasa al siguiente vecino. Estando el objeto dentro de la región de búsqueda, si la función `ESSKYLINENOESPACIAL` devuelve un valor verdadero (Líneas 39-40), entonces se agrega el objeto a la pila `h`, sólo para revisar luego sus vecinos. Si devuelve falso (Líneas 41-49), se obtiene la celda Voronoi del objeto que se utilizará en la evaluación del *skyline*.

Con respecto al *VS²* original, se realizó una variación en este punto de ejecución del algoritmo (Líneas 43-45), se verifica la pertenencia del objeto al *convex hull* o en su defecto se identifica si existe una intersección con la celda de Voronoi del objeto. Estas verificaciones se agregaron para reducir el número de comparaciones de dominancia innecesarias si el objeto ya pertenece a `ch`. Si el objeto pertenece, lo etiqueta con el procedimiento `MARCA_SKYLINE_CONVEX_HULL` y lo inserta en la pila `h` para que posteriormente pueda ser tomado como *skyline*.

Si el objeto no pertenece al *convex hull* y no existen dimensiones no espaciales (Líneas 46-48), se verifica si la celda de Voronoi a la que pertenece es dominada por el **skyline** y la pila **hS**.

Para las comparaciones de dominancia, se utilizan como dimensiones: las distancias mínimas de la celda a los puntos de referencia almacenadas en **distanciasVoronoi** y como dimensiones no espaciales, las que posee el objeto. Si la celda no es dominada, se incluye el objeto en la pila **h** para continuar con el siguiente vecino.

Luego de finalizar la exploración de los objetos en la pila **h**, el algoritmo itera la pila **hS** para verificar si cada uno de los objetos es dominado por los que ya están en la lista **skyline** (Líneas 54-56); si no es dominado, lo inserta. Al vaciarse la pila **hS**, culmina el algoritmo devolviendo el conjunto **skyline**.

Algoritmo 5 VC2S+ (Parte 1)

```

1: Integer iPtoDin, dNoEsp
2: List<Object> candidatos, skyline, skylineNoEsp, caldasVoronoi, distanciasVoronoi, ptosRef, attObjs
3: List<List<Object>> vecinosVoronoi
4: DelaunayGraph grafo, ConvexHull ch, Rectangle regionBusqueda

5: procedure COMENZAR(aConf, aPtosRef, aDimNoEsp)      ▷ Calcula el primer Skyline (Modo inicial)
6:   INICIALIZAR(aConf, aPtosRef, aDimNoEsp)
7:   skylineNoEsp  $\leftarrow \emptyset
8:   grafo  $\leftarrow$  CREARDELAUNAYGRAPH(ptosRef, candidatos)      ▷ Grafo (http://code.google.com/p/jdt)
9:   ch  $\leftarrow$  CREARCONVEXHULL(ptosRef)                      ▷ Crea el Convex Hull. Algoritmo: Graham Scan
10:  if dNoEsp  $> 0 then                                ▷ Compara solo dimensiones no espaciales, arma region de busqueda
11:    CALCULARSKYLINENOESPACIAL(skylineNoEsp, candidatos, regionBusqueda) ▷ SkylineNoEsp
12:  end if
13:  skyline  $\leftarrow$  GENERARVS2(skyline, candidatos, grafo, ch)          ▷ Construye el skyline
14:  ADDALL(skyline, skylineNoEsp)           ▷ Agrega los objetos skylineNoEsp que no estaban en skyline
15:  return skyline
16: end procedure

17: procedure INICIALIZAR(aConf, aPtosRef, aDimNoEsp)      ▷ Lee los datos de los archivos de entrada
18:   attObjs, skyline  $\leftarrow \emptyset
19:   ptosRef  $\leftarrow$  OBTENERPUNTOSREFERENCIA(aPtosRef, iPtoDin)
20:   candidatos  $\leftarrow$  ASIGNARCANDIDATOS(aConf)      ▷ Almacena los disponibles en la lista de candidatos
21:   attObjs  $\leftarrow$  OBTENERVALORESDIMENSIONESNOESPACIALES(aDimNoEsp, dNoEsp)
22: end procedure$$$ 
```

Algoritmo 6 VC2S+ (Parte 2)

```

23: procedure GENERARVS2(skyline, candidatos, grafo, ch)                                ▷ Algoritmo VS2
24:   Object obj  $\leftarrow$  OBTENERNN(ptosRef[iPtoDin])                                     ▷ Encuentra objeto mas cercano al punto móvil
25:   MinHeap h  $\leftarrow \{obj, ADIST(obj, ch, attObjs)\}$                                          ▷ Agrega el objeto a la pila
26:   List visitados  $\leftarrow \{obj\}$ 
27:   MinHeap hS  $\leftarrow \emptyset$                                                        ▷ Pila donde se colocara el skyline
28:   while not ISEMPTY(h) do                                                 ▷ Recorre el Grafo Delaunay partiendo con obj
29:     (p, d)  $\leftarrow$  REMOVEFIRST(h)
30:     if not ES SKYLINE NO ESPACIAL(p) then
31:       if ESTA MARCADO SKYLINE CONVEXHULL(p) then AGREGAR SKYLINE(skyline, p)
32:       else if not ES DOMINADO(p, skyline, hS) then AGREGAR PILA SKYLINE(hS, (p, d))
33:     end if
34:     List<Object> vecinos  $\leftarrow$  OBTENER VECINOS VC(grafo, vecinosVoronoi, p)
35:     for v  $\in$  vecinos do
36:       if not ESTA VISITADO(visitados, v) then
37:         AGREGAR VISITADOS(visitados, v)
38:         if dNoEsp = 0 or ( dNoEsp > 0 and ESTA REGION BUSQUEDA(v) ) then
39:           if ES SKYLINE NO ESPACIAL(v) then
40:             AGREGAR PILA(h, (v, ADIST(v, ch, attObjs)))      ▷ Agrega v a h, revisará sus vecinos
41:           else
42:             vc  $\leftarrow$  OBTENER VC(celdasVoronoi, v)          ▷ Agregado para reducir comparaciones
43:             if ESTA CONVEXHULL(ch, v) or HAY INTERSECCION(vc, ch) then
44:               MARCA SKYLINE CONVEXHULL(v)
45:               AGREGAR PILA(h, (v, ADIST(v, ch, attObjs)))
46:             else if (dNoEsp = 0 and not CELDA ES DOMINADA(vc, skyline, hS) ) or dNoEsp > 0 then
47:               AGREGAR PILA(h, (v, ADIST(v, ch, attObjs)))
48:             end if
49:           end if
50:         end if
51:       end if
52:     end for
53:   end while
54:   while not ISEMPTY(hS) do                                              ▷ Última validacion skyline, agrega no dominados
55:     (p, d)  $\leftarrow$  REMOVEFIRST(hS) if not ES DOMINADO(p, skyline) then AGREGAR SKYLINE(skyline, p)
56:   end while
57:   return skyline                                                               ▷ Conjunto skyline construido
58: end procedure

```

Este algoritmo únicamente agrega al *skyline*, si hay certeza de que el objeto no será dominado por otro, de lo contrario lo mantiene en la pila de posibles objetos pertenecientes al *skyline*, *hS*. La ventaja es que ningún objeto tendrá que ser eliminado del *skyline*. La desventaja es que en el peor de los casos, se realizarán comparaciones adicionales para un mismo objeto. Por ejemplo: para insertar el objeto en *h* se debe verificar si la celda Voronoi de ese objeto no es dominada por los objetos que están en *skyline* y *hS*; para insertar el objeto en *hS* se debe verificar si el objeto no es dominado por los que ya están en *skyline* y *hS*; para insertar el objeto en *skyline*, se debe verificar si el objeto no es dominado por los que ya están en *skyline*.

3.3.3.2 Modo actualización de estatus

En cuanto al *modo actualización de estatus*, se iniciará ejecutando el procedimiento CALCULARSOLUCIONCAMBIOSTATUS (Algoritmo 5, p. 63, Líneas 59-72) que reinicia las estructuras `skyline`, `skylineNoEsp` `vecinosVoronoi`, `celdasVoronoi` y `distanciasVoronoi` para prepararse a los cambios de los objetos. Los nuevos objetos ocupados son eliminados de `candidatos` (Líneas 61-63) y los nuevos disponibles son agregados (Líneas 64-66).

Tomando la lista `candidatos` y si en los datos existen dimensiones no espaciales, se realizan las comparaciones para extraer el *skyline* no espacial y calcular de nuevo la región de búsqueda (Línea 67).

Posteriormente, se construye de nuevo el grafo *Delaunay* con los candidatos resultantes del cambio de estatus (Línea 68). Una vez creado el grafo, se genera el *skyline* espacial desde el inicio llamando al procedimiento GENERARVS2 (Línea 69).

Algoritmo 7 VC2S+ (Parte 3)

```

59: procedure CALCULARSOLUCIONCAMBIESTATUS(ocupados, disponibles)
60:   vecinosVoronoi, celdasVoronoi, distanciasVoronoi, skyline, skylineNoEsp  $\leftarrow \emptyset$ 
61:   for o  $\in$  ocupados do                                 $\triangleright$  Elimina los nuevos objetos ocupados
62:     ELIMINARCANDIDATO(candidatos, o)
63:   end for
64:   for d  $\in$  disponibles do                       $\triangleright$  Agrega los nuevos objetos disponibles
65:     AGREGARCANDIDATO(candidatos, d)
66:   end for
67:   CALCULARSKYLINENOESPECIAL(skylineNoEsp, candidatos, regionBusqueda)
68:   grafo  $\leftarrow$  CREARDELAUNAYGRAPH(ptosRef, candidatos)
69:   GENERARVS2(skyline, candidatos, grafo, ch)           $\triangleright$  Construye el skyline
70:   ADDALL(skyline, skylineNoEsp)
71:   return skyline
72: end procedure

```

3.3.3.3 Modo actualización de desplazamiento

El *modo actualización de desplazamiento* inicia con el procedimiento CALCULARSOLUCIONDESPLAZAMIENTO (Líneas 73-84), donde se almacena el *convex hull* anterior en el polígono `chPrev` (Línea 74) y se crea el *convex hull* alterado por el desplazamiento, `ch` (Línea 75). Además, se actualiza la nueva posición del punto de referencia móvil y en la

lista `attObj`, las distancias de cada candidato a esa nueva posición (Línea 76). Tomando los patrones de cambio descritos en la sección 3.1.3.3 y los polígonos `chPrev` y `ch` (Línea 77), se determina el caso que aplique.

Para el patrón I, se devuelve el mismo conjunto *skyline* (Línea 78). Si existen dimensiones no espaciales, será necesario calcular la nueva región de búsqueda ya que ha sido alterada por el desplazamiento (Línea 79). Para el patrón VI, es necesario ejecutar el algoritmo *VS*² desde el inicio, invocando el procedimiento `GENERARVS2` (Línea 80). Para los patrones II, III, IV y V (Líneas 81-83), se actualiza el *skyline* ejecutando el procedimiento `GENERARVCS2`. Una vez actualizado el *skyline* espacial, se retorna el *skyline*.

El procedimiento `GENERARVCS2` (Algoritmo 9, p. 67, Líneas 85-131), es similar a `GENERARVS2` con la diferencia que se mantiene el *skyline* previo y se calcula el sector sin cambio (Línea 89), el cual es la zona que no es afectada por el desplazamiento del punto de referencia y permite ignorar objetos irrelevantes. Este sector es el polígono verde que se observa en las Figuras 3.10 y 3.13. Como es necesario actualizar directamente la lista `skyline`, los objetos son agregados o eliminados según lo indique la función de verificación de dominancia (Líneas 92-99). Al vaciarse la pila `h`, se verifica el *skyline* completo para eliminar los objetos que no fueron detectados antes (Líneas 122-129). Para este paso, se realizó la misma modificación que en `GENERARVS2` (Líneas 124-125).

Algoritmo 8 VC2S+ (Parte 4)

▷ Modo actualización de desplazamiento

```

73: procedure CALCULARSOLUCIONDESPLAZAMIENTO(nuevaPosPtoRef)
74:   chPrev  $\leftarrow$  ch
75:   ch  $\leftarrow$  ACTUALIZARCONVEXHULL(ptosRef, nuevaPosPtoRef)      ▷ Convex hull con nueva posicion
76:   ACTUALIZARDISTANCIAPTODIN(candidatos, attObj, nuevaPosPtoRef)    ▷ Distancia pto móvil
77:   caso  $\leftarrow$  IDENTIFICARCASODESPLAZAMIENTO(ch, chPrev)
78:   if caso = 1 then return skyline                                ▷ No se realizan cambios en el skyline
79:   if dNoEsp > 0 then regionBusqueda  $\leftarrow$  OBTENERNUEVARREGIONBUSQUEDA(skylineNoEsp)
80:   if caso = 6 then return GENERARVS2(skyline, candidatos, grafo, ch)    ▷ Construye el skyline
81:   GENERARVCS2(skyline, candidatos, grafo, ch)                      ▷ Casos 2-5. Actualiza el skyline
82:   ADDALL(skyline, skylineNoEsp)
83:   return skyline
84: end procedure
```

Algoritmo 9 VC2S+ (Parte 5)

```

85: procedure GENERARVCS2(skyline, candidatos, grafo, ch)                                ▷ Algoritmo VCS2
86:   Object obj  $\leftarrow$  OBTENERNN(ptosRef[iPtoDin])
87:   MinHeap h  $\leftarrow \{obj, \text{ADIST}(obj, ch, attObjs)\}$ 
88:   List visitados  $\leftarrow \{obj\}$ 
89:   Polygon sectorSinCambio  $\leftarrow$  GENERARSECTORSCAMBIOSINCAMBIO(ch, chPrev)
90:   while not ISEMPTY(h) do
91:     (p, d)  $\leftarrow$  REMOVEFIRST(hS)
92:     if not ES_SKYLINE_NO_ESPACIAL(p) then
93:       if ESTAMARCADO_SKYLINE_CONVEXHULL(p) then
94:         if not ESTASKYLINE(p) then AGREGARSKYLINE(skyline, p)
95:         else
96:           if not (ESDOMINADO(p, skyline) or ESTASKYLINE(p)) then AGREGARSKYLINE(skyline, p)
97:           else if ESDOMINADO(p, skyline) and ESTASKYLINE(p) then ELIMINARSKYLINE(skyline, p)
98:         end if
99:       end if
100:      List<Object> vecinos  $\leftarrow$  OBTENERVECINOSVC(grafo, vecinosVoronoi, p)
101:      for v  $\in$  vecinos do
102:        if not ESTAVISITADO(visitados, v) then
103:          AGREGARVISITADOS(visitados, v)
104:          if ESTASECTORSCAMBIOSINCAMBIO(sectorSinCambio, v) then
105:            continue
106:          else if dNoEsp = 0 or (dNoEsp > 0 and ESTAREGIONBUSQUEDA(v) then
107:            if ES_SKYLINE_NO_ESPACIAL(v) then ▷ Agrega v a h para revisar luego sus vecinos
108:              AGREGARPILA(h, (v, ADIST(v, ch, attObjs)))
109:            else
110:              vc  $\leftarrow$  OBTENERVC(celdasVoronoi, v) ▷ Agregado para reducir comparaciones
111:              if ESTACONVEXHULL(ch, v) or HAYINTERSECCION(vc, ch) then
112:                MARCA_SKYLINE_CONVEXHULL(v)
113:                AGREGARPILA(h, (v, ADIST(v, ch, attObjs)))
114:              else if (dNoEsp = 0 and not CELDA_ESDOMINADA(vc, skyline, hS)) or dNoEsp > 0 then
115:                AGREGARPILA(h, (v, ADIST(v, ch, attObjs)))
116:              end if
117:            end if
118:          end if
119:        end if
120:      end for
121:    end while
122:    while not ISEMPTY(skyline) do ▷ Última validacion skyline, elimina dominados
123:      p  $\leftarrow$  GETNEXT(skyline)
124:      if ES_SKYLINE_NO_ESPACIAL(p) or (ESTAMARCADO_SKYLINE_CONVEXHULL(p) and ESTACONVEXHULL(ch, p)) then
125:        continue
126:      else if ESDOMINADO(p, skyline) then
127:        ELIMINARSKYLINE(skyline, p)
128:      end if
129:    end while
130:    return skyline ▷ Conjunto skyline actualizado
131:  end procedure

```

3.4 Solución propuesta

Los algoritmos 2B2S y IB2S al momento de realizar la búsqueda de candidatos para obtener el *skyline*, chequean la disponibilidad de todos los objetos en el plano y posteriormente realizan las comparaciones de dominancia, tomando el conjunto completo de estos candidatos.

El algoritmo VC2S+ poda el espacio de búsqueda de los candidatos en el momento en que la celda Voronoi del objeto es dominada por el *skyline* espacial o por los elementos de la pila o si el objeto pertenece al sector que no presenta cambios. También, elimina las verificaciones de dominancia para los objetos que están dentro del *convex hull*. Sin embargo, la representación del plano es estática, como consecuencia, debe crear nuevamente el grafo *Delaunay* cada vez que existan cambios de estatus en los objetos.

En contraste, la solución propuesta denominada *Continuous Dynamic Spatial Skyline* (CD2S) (Algoritmos 10-12, , pp. 74, 78 y 81), utiliza la misma representación del plano que tienen los algoritmos 2B2S y IB2S, para garantizar que la actualización continua del estatus de los objetos no necesite la reconstrucción de la estructura. Ademas, utiliza regiones de búsqueda para podar la exploración de candidatos. Similar al VC2S+, el CD2S utiliza el *convex hull* para reducir las comparaciones de dominancia y permitir la obtención temprana de algunos resultados.

3.4.1 Ejemplo de ejecución del algoritmo CD2S

Utilizando las mismas simulaciones para los ejemplos descritos en la sección 3.1.2, se explica el algoritmo CD2S.

3.4.1.1 Ejemplo I: Dimensiones sólo espaciales

En el *modo inicial*, inicialmente se tiene una región de búsqueda que cubre el plano y se afina con cada elemento *skyline* que se obtenga, región gris en la Figura 3.14, se obtiene el *convex hull* completo y el *convex hull* estático con los cuales se obtiene el *skyline* estático y dinámico (*skylineEst* y *SkylineDin*), en este caso todos son estáticos. Con ellos se actualiza

la región de búsqueda. Luego, tomando los objetos fuera del *convex hull* y dentro de la región de búsqueda revisa el plano para obtener los candidatos al *skyline*. En base al ejemplo de la Figura 3.14, el algoritmo retorna 14 puestos interesantes, de los cuales cinco pertenecen al *convex hull* estático.

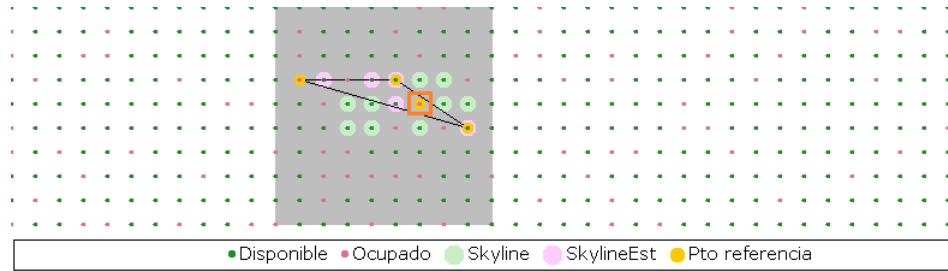


Figura 3.14: Ejemplo I, *modo inicial* (simulación 0)

El algoritmo en el *modo actualización de estatus*, elimina de los candidatos y del *skyline*, los puestos disponibles que pasaron a estar ocupados y, los que pasaron a estar disponibles, si están dentro del *convex hull* (dos puestos), los agrega directamente al *skyline* y con los restantes realiza comparaciones de dominancia con los puestos que aún pertenecen al *skyline*.

Debido a que se eliminaron objetos del *skyline* (11 puestos), la región de búsqueda cambió, en este caso se agranda ligeramente, se obtienen esos candidatos y con esos y los existentes, se realizan comparaciones de dominancia para actualizar el *skyline* (este paso, incluye un nuevo puesto). El algoritmo CD2S retorna en la Figura 3.15, el *skyline*, donde cuatro de los ocho puestos pertenecen al *skyline* estático.

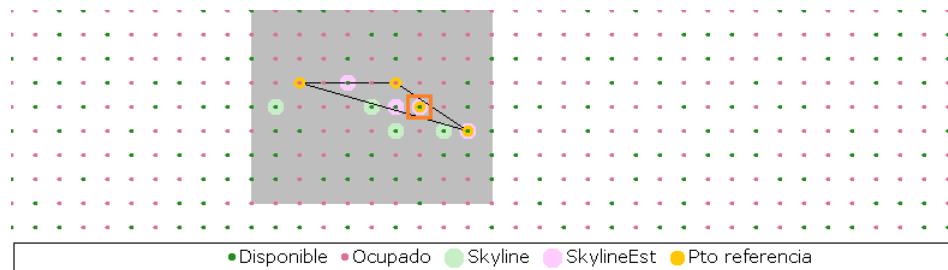


Figura 3.15: Ejemplo I, *modo actualización de estatus* (simulación 1)

El algoritmo en el *modo actualización de desplazamiento*, mantiene el *skyline* estático. Este procedimiento, identifica el mismo patrón de desplazamiento que VC2S+, en este caso, se revisan los puestos que forman parte del *skyline* para reasignar los que estén dentro del nuevo *convex hull* al *skyline* dinámico (tres puestos), marcado con círculos azules en la

Figura 3.16; luego obtiene los nuevos objetos que están dentro del *convex hull* y los agrega al *skyline* dinámico (cinco puestos). Finalmente, actualiza la región de búsqueda y revisa la dominancia de aquellos que estén fuera del *convex hull* y del sector sin cambio (polígono verde) y dentro de la región de búsqueda para agregarlos al *skyline* (cuatro puestos). Se retorna el *skyline* compuesto por: cuatro puestos pertenecientes al *convex hull* estático, ocho al *convex hull* dinámico y cinco fuera del *convex hull*.

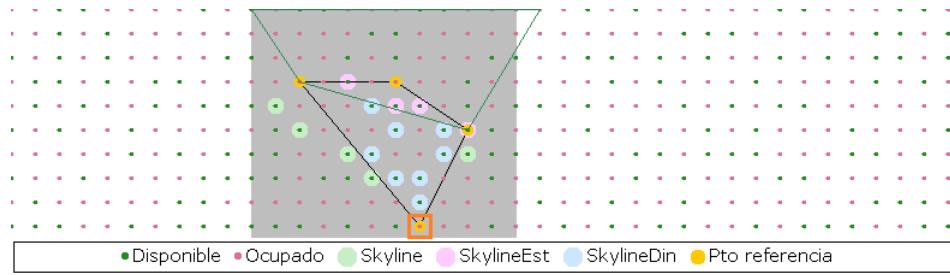


Figura 3.16: Ejemplo I, *modo actualización de desplazamiento* (simulación 2)

3.4.1.2 Ejemplo II: Dimensiones espaciales y no espaciales

Al igual que el algoritmo VC2S+, en presencia de dimensiones no espaciales, en el *modo inicial*, el algoritmo calcula el *skyline* no espacial y genera una región de búsqueda. Posteriormente, extrae los candidatos disponibles, en forma similar al proceso anterior. El algoritmo CD2S, retorna en la Figura 3.17, como *skyline* no espacial, 10 de los 70 puestos marcados como interesantes.

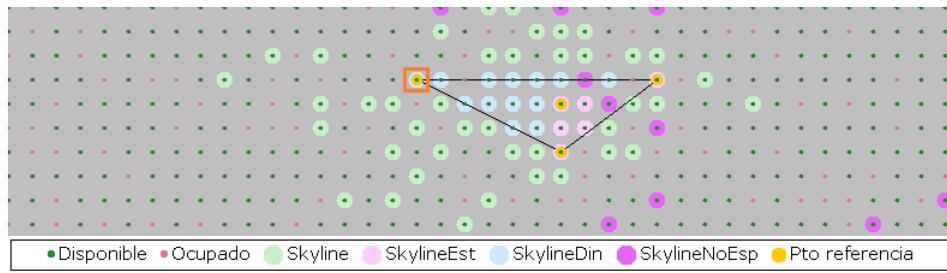


Figura 3.17: Ejemplo II, *modo inicial* (simulación 0)

Para el *modo actualización de estatus* se eliminan los puestos que ahora están ocupados, se agregan los nuevos puestos disponibles: directamente al *skyline* si están dentro del *convex hull* (un puesto) y se eliminan los dominados por estos espacialmente; para los que están fuera se chequea dominancia espacial con respecto al *skyline*. Luego se revisan los nuevos puestos y objetos dominados espacialmente si forman parte del *skyline* no espacial, verificando sólo esas dimensiones (se agrega un puesto). Se actualiza la región

de búsqueda con los que están en el *skyline* no espacial, debido a que la nueva región es menor, se verifican la dominancia (espacial y no espacial) de los puestos que siguen estando disponibles para completar el *skyline*. Se devuelve como *skyline* en la Figura 3.18: tres puestos pertenecientes al *skyline* no espacial, uno al *convex hull* estático, cinco al dinámico y 43 fuera del *convex hull*.

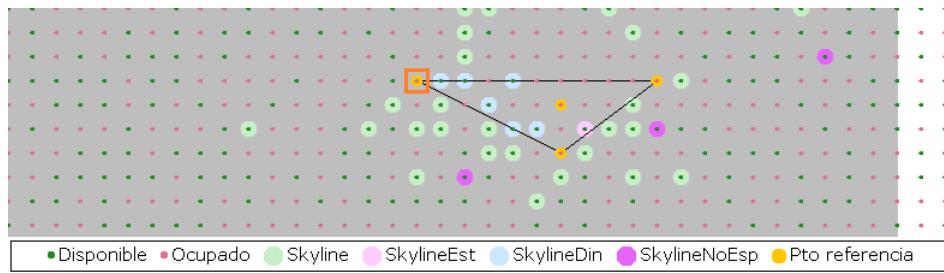


Figura 3.18: Ejemplo II, *modo actualización de estatus* (simulación 1)

En la simulación de desplazamiento, se mantiene el *skyline* estático y el *skyline* no espacial, pero se calcula con este último, la nueva región de búsqueda ya que cambia porque el punto de referencia móvil se desplazó. El patrón de cambio se identifica, en este caso es el III, donde se revisan los puestos existentes en el *skyline* para eliminar los que ahora son dominados, al igual que el VC2S+ revisa sólo la región fuera del polígono verde y del *convex hull* y dentro de la región de búsqueda. Se actualiza la distancia al punto de referencia móvil de los objetos candidatos y los pertenecientes al *skyline*. Se devuelve como *skyline* en la Figura 3.19: tres puestos del *skyline* no espacial, uno perteneciente al *convex hull* estático, dos al *convex hull* dinámico y 21 fuera del *convex hull*.

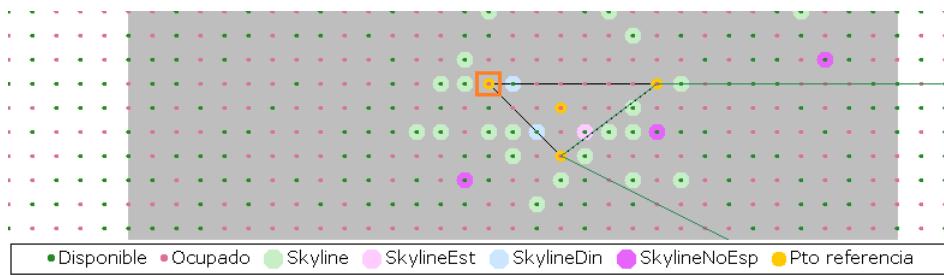


Figura 3.19: Ejemplo II, *modo actualización de desplazamiento* (simulación 2)

3.4.2 Estructuras utilizadas por el algoritmo CD2S

Adicionalmente a las estructuras empleadas en los algoritmos 2B2S y 1B2S, se utiliza la variable `regionBusqueda` del tipo rectángulo que será utilizada para restringir la búsqueda de candidatos, visible como la región gris en la Figura 3.14 y se emplean las variables `ch` y

`chEst` del tipo polígono para almacenar el *convex hull* completo y el resultante al considerar sólo los puntos de referencia estáticos. En las figuras se observa `ch` como el polígono negro y aunque `chEst` no está marcado explícitamente, es el que encierra a los círculos *skyline* de color rosado.

Se tienen varias listas para indexar los diferentes tipos de *skyline*: la primera, `skylineNoEsp` para referenciar a los objetos pertenecientes al *skyline* de las dimensiones no espaciales (círculos magenta), la lista `skylineCHest` para almacenar las referencias a los objetos pertenecientes al polígono `chEst` (círculos rosados) y `skylineCHdin` para los que están fuera de esa región y dentro del `ch` (círculos azules).

Esta separación del *skyline* en diversas listas se realizó, debido a que se observó que los objetos pertenecientes al `chEst` siempre que estén disponibles formarán parte del *skyline* sin importar como haya sido el desplazamiento del punto de referencia móvil, entonces al igual que los objetos en la lista `skylineNoEsp`, los objetos en la lista `skylineCHest` sólo se verán afectados con algún cambio de disponibilidad, ameritando únicamente una eliminación o agregación en esta última lista si el objeto que cambio de disponibilidad pertenece al polígono `chEst`.

3.4.3 Descripción del algoritmo CD2S

A continuación se describe el algoritmo CD2S por modo de ejecución.

3.4.3.1 Modo inicial

El algoritmo CD2S inicia su ejecución en *modo inicial* con el procedimiento COMENZAR (Algoritmo 10, p. 74, Líneas 6-18). Al principio, al igual que los algoritmos 2B2S y IB2S, invoca el procedimiento INICIALIZAR (Línea 7). Posteriormente, crea el *convex hull* `ch` formado por todos los puntos de referencia (Línea 8) y, tomando sólo los puntos de referencia estáticos genera el *convex hull* `chEst` (Línea 9). Se construye la región de búsqueda (Línea 10), que inicialmente abarcará todo el plano y luego se afinará para evitar la búsqueda exhaustiva de objetos.

Luego, mediante el procedimiento OBTENERCANDIDATOSREGION (Línea 11), utilizando el *convex hull* y la región de búsqueda, encuentra aquellos objetos disponibles para ser agregados a la lista **candidatos**. Este procedimiento (Líneas 19-23), invoca a la función OBTENERMINRECTANGULO (Línea 20) que dado el polígono **ch** permitirá conseguir el mínimo rectángulo que lo encierra, **regionBusquedaTmp**, el cual se utilizará para encontrar los candidatos en la región del *convex hull*.

La búsqueda se realiza a través de REVISARSUBREGION (Línea 21), que partiendo de la variable **regionBusquedaTmp** permite identificar los objetos que están disponibles. Los objetos contiguos se exploran mediante particiones cuya longitud variará desde un *byte* (8 objetos) hasta un *long* (64 objetos), dependiendo del tamaño del espacio de búsqueda. Estas particiones permiten descartar rápidamente espacios ocupados porque se reflejarán como cero en el valor de la partición al realizar el chequeo, lo que evitara revisiones innecesarias de objeto por objeto si están todas ocupadas.

Para aquellos *bits* que estén encendidos, se calcularán las distancias del objeto a cada punto de referencia (si no han sido calculadas antes), sino, se actualiza sólo la distancia correspondiente al punto en desplazamiento. Además, se verifica si el objeto pertenece al **ch** o al **chEst** para agregar directamente la referencia del objeto a la lista **skyline** sin realizar evaluaciones de dominancia y a la lista **skylineCHest** o **skylineCHdin** según sea el caso. Si no pertenece al polígono **ch**, sólo se agregará a la lista **candidatos**.

Si las dimensiones son sólo espaciales, entonces a medida que se agreguen objetos a la lista **skyline**, se actualizará la variable **regionBusqueda** cuya región se irá reduciendo (Apéndice C, Figura C.1(b)). Similar al *VCS²*, se calcula la región de visibilidad del *convex hull* con respecto al objeto *skyline* (Ej: *regionBusqueda(s₁)*), pero ésta se intersecta con la región de la iteración anterior (Ej: *regionBusqueda(s₁) ∩ regionBusqueda(s₂)*). Este método es el utilizado en el algoritmo *B²S²* [1] para limitar el espacio de búsqueda del *skyline* espacial. Luego, se realiza la búsqueda de candidatos que estén fuera del mínimo rectángulo que encierra al polígono **ch** y dentro del rectángulo **regionBusqueda** utilizando el procedimiento REVISARALREDEDORESREGION (Línea 22).

Posteriormente en el procedimiento COMENZAR, en caso de existir alguna dimensión no espacial (Líneas 12-14), se llena la lista `skylineNoEsp`, utilizando el algoritmo SFS [22] que ordena los objetos disponibles encontrados por las dimensiones no espaciales y chequea la dominancia sólo en estas dimensiones. Con cada objeto agregado, se actualiza la región de búsqueda. Finalmente, se obtienen los objetos restantes del *skyline*, utilizando la función CALCULARSKYLINE (Línea 15), donde se itera cada objeto en la lista `candidatos` revisando su dominancia contra los que ya están en las listas `skylineNoEsp` y `skyline`. Se supone que si un objeto *skyline* está marcado como perteneciente a la lista `skylineCHdin` o `skylineCHest` no será dominado por algún candidato debido a que ya se tiene la certeza que el objeto pertenece *skyline*. Por lo tanto, si el algoritmo identifica que el candidato es incomparable empleando sólo las dimensiones espaciales, no será necesario verificar las no espaciales. Por último, es retornado el *skyline* (Líneas 16-17).

Algoritmo 10 CD2S (Parte 1)

```

1: Integer iPtoDin, dNoEsp
2: Byte[ ][] plano
3: List<Object> candidatos, skyline, skylineCHdin, skylineCHest, skylineNoEsp, ptosRef, attObjs
4: ConvexHull ch, chEst
5: Rectangle regionBusqueda

6: procedure COMENZAR(aConf, aPtosRef, aDimNoEsp)                                ▷ Modo inicial
7:   INICIALIZAR(aConf, aPtosRef, aDimNoEsp)                                         ▷ Misma funcion que 2B2S y IB2S
8:   ch  $\leftarrow$  CREARCONVEXHULL(ptosRef)                                              ▷ Crea el Convex Hull. Algoritmo: Graham Scan
9:   chEst  $\leftarrow$  OBTENERCONVEXHULLESTATICO(ptosRef)                                     ▷ Toma solo los puntos de referencia estaticos
10:  regionBusqueda  $\leftarrow$  CREARRECTANGULO(0, nColReales, 0, nFil)                      ▷ Inicialmente el plano
11:  OBTENERCANDIDATOSREGION(regionBusqueda, ch, chEst, skyline, skylineCHest,
    skylineCHdin, plano, attsObj, candidatos)                                         ▷ Agrega los disponibles dentro de la region
12:  if dNoEsp > 0 then                                                 ▷ Compara solo dimensiones no espaciales, arma region de busqueda
13:    CALCULARSKYLINENOESPACIAL(skylineNoEsp, candidatos, regionBusqueda)
14:  end if
15:  CALCULARSKYLINE(skyline, candidatos)                                         ▷ Actualiza el skyline, revisa la dominancia de los candidatos
16:  ADDALL(skyline, skylineNoEsp)
17:  return skyline
18: end procedure                                                               ▷ Obtiene los candidatos al skyline y los que forman parte de: skylineCHest, skylineCHdin

19: procedure OBTENERCANDIDATOSREGION(regionBusqueda, ch, chEst, skyline, skylineCHest,
    skylineCHdin, plano, attsObj, candidatos)
20:   Rectangle regionTmp  $\leftarrow$  OBTENERMINRECTANGULO(ch)
21:   REVISARSUBREGION(regionTmp, regionBusqueda, ch, chEst, skyline, skylineCHest,
    skylineCHdin, plano, attsObj, candidatos)
22:   REVISARALREDEDORESSUBREGION(regionTmp, regionBusqueda, plano, attsObj, candidatos)
23: end procedure

```

3.4.3.2 Modo actualización de estatus

Para el *modo actualización de estatus*, se ejecuta CALCULARSOLUCIONCAMBIOESTATUS (Algoritmo 11, p. 78, Líneas 24-74). Inicialmente, graba la región de búsqueda de la ejecución anterior en la variable `regionPrev` (Línea 25) y similar al algoritmo IB2S, actualiza el plano (Línea 28).

El algoritmo, itera la lista `ocupados` (Líneas 29-33) para eliminar los elementos de `candidatos` y si lo amerita también de las listas `skyline` y `skylineNoEsp`.

Si no se eliminaron objetos del `skyline` (Líneas 34-36), no será necesario revisar los candidatos que ya estaban antes del cambio de estatus, por lo que se almacenan temporalmente en `candidatosTmp` y se vacía la lista `candidatos`. Sino (Líneas 37-43), se reconstruye la región de búsqueda con los objetos que quedaron en las listas `skyline` o `skylineNoEsp`.

Se itera la lista `disponibles` (Líneas 44-50) y si existen dimensiones no espaciales (Línea 45), se agrega el nuevo objeto disponible a la lista temporal `candidatosNoEspNvos` para ser analizado luego.

Posterior a la inserción de los disponibles, se evalúa si el objeto pertenece a la región de búsqueda. De ser así (Líneas 46-49), se verifica si el objeto está en el polígono `ch` o `chEst` para insertarlo a la lista `skyline` y a su respectiva lista `skylineCHdin` o `skylineCHest` (Línea 47); sino está dentro del polígono `ch`, lo añade al índice temporal `candidatosNvos`. El algoritmo consulta si el nuevo objeto domina espacialmente a algún otro objeto del `skyline` o si es incomparable, para agregarlo y sino, desecharlo (Línea 48).

Es importante destacar que para reducir el número de comparaciones innecesarias, se asume que los objetos pertenecientes al polígono `ch` no pueden ser dominados por otro.

Si existen dimensiones no espaciales (Líneas 51-62), se determina el caso a seguir:

- * Caso 1: si no se eliminaron objetos de la lista `skylinoNoEsp` y se agregaron objetos a la lista `candidatos`.
- * Caso 2: si se eliminaron objetos de la lista `skylinoNoEsp` y se agregaron objetos a la lista `candidatos`.
- * Caso 3: si se eliminaron objetos de la lista `skylinoNoEsp` y no se agregaron objetos a la lista `candidatos`.

Para el caso 1 (Líneas 51-54), se ejecuta el procedimiento `AGREGARDESCARTADOSSKYLINENOESP` (Línea 52), que inserta en la lista temporal `candidatosNoEspNvos` aquellos objetos que fueron eliminados de la lista `skyline` en la verificación de dominancia espacial, ya que aún podrían formar parte de la lista `skylineNoEsp`. Después, ordena los elementos en la lista `candidatosNoEspNvos` primero por las dimensiones no espaciales y luego por las demás y actualiza la lista `skylineNoEsp` (Línea 53).

Para el caso 2 (Líneas 51-58), se agregan los objetos de la lista `candidatosNoEspNvos` a la lista `candidatosNoEsp`, la cual contiene todos los candidatos al *skyline* no espacial (Línea 56). Los elementos de la lista `candidatosNoEsp`, se ordenan y con ellos se inicia la revisión del *skyline* no espacial, partiendo desde el mínimo entre el menor índice de los objetos eliminados y el menor índice de los objetos agregados; el índice `i` de partida es dado por `OBTENERÍNDICEREVISIÓN` (Línea 57). Se comienza a partir de la posición `i` para evitar revisiones innecesarias de los candidatos, ya que al estar ordenados por sus dimensiones, los objetos que estén antes de `i` no han sido afectados por los cambios ocurridos.

Para el caso 3 (Líneas 59-62), se ordena la lista `candidatosNoEsp`, se obtiene el menor índice sólo de los objetos eliminados (Línea 60) y, al igual que el caso 2, actualiza la lista `skylineNoEsp` (Línea 61).

Por otro lado, si se eliminaron objetos del *skyline* por el cambio de estatus, es posible que la región de búsqueda haya sido alterada aún cuando el polígono `ch` no ha cambiado. Por lo tanto, se deben encontrar los objetos disponibles que estén en la nueva región `regionBusqueda` y que no estaban antes como candidatos. Para esto, se invoca al procedimiento `BUSCARNVCANDIDATOS` con la región de búsqueda anterior, `regionPrev` y la actual (Línea 64).

Finalmente, se actualiza el *skyline* (Línea 66) revisando la dominancia de los candidatos encontrados (excluyendo los que se agregaron a `candidatosNvos` ya que fueron revisados antes).

Esta solución, tiene como ventaja que la estructura base para los objetos no necesita ser reconstruida cada vez que exista un cambio en la disponibilidad de estas como lo hace VC2S+. Esto permite aprovechar las propiedades del *skyline* espacial en la actualización del conjunto *skyline* a causa de las variaciones en la disponibilidad de los objetos, lo cual evita que *skyline* deba ser calculado desde el principio.

A diferencia del algoritmo VC2S+, un cambio en la disponibilidad de un objeto implicaría un simple acceso a la matriz para modificar su estatus y luego de ser necesario, se actualizaría el conjunto *skyline*.

Algoritmo 11 CD2S (Parte 2)

▷ Modo actualización de estatus

```

24: procedure CALCULARSOLUCIONCAMBIOESTATUS(ocupados, disponibles)
25:   Rectangle regionPrev  $\leftarrow$  regionBusqueda
26:   List<Object> candidatosTmp, candidatosNoEspNvos, candidatosNvos  $\leftarrow$   $\emptyset$ 
27:   Integer elimSkyline, elimSkylineNoEsp, agrCandidatos, agrCandidatosNoEsp  $\leftarrow$  0           ▷ Contadores
28:   ACTUALIZARSTATUS(plano, ocupados, disponibles)
29:   for o  $\in$  ocupados do                                ▷ Elimina del skyline y candidatos, los nuevos objetos ocupados
30:     if o  $\in$  skyline then ELIMINAROBJSKYLINE(skyline, skylineCHdin, skylineCHest, o, elimSkyline)
31:     if o  $\in$  skylineNoEsp then ELIMINAROBJSKYLINENOESP(skylineNoEsp, o, elimSkylineNoEsp)
32:     ELIMINARCANDIDATO(candidatos, o)
33:   end for
34:   if elimSkyline = 0 then                      ▷ No hay eliminados del skyline, solo se revisará nuevos candidatos
35:     candidatosTmp  $\leftarrow$  candidatos
36:     candidatos  $\leftarrow$   $\emptyset$ 
37:   else
38:     if dNoEsp = 0 then                      ▷ Hay eliminados del skyline, región podría ser mayor
39:       RECALCULARREGIONBUSQUEDA(skyline, regionBusqueda)
40:     else if elimSkylineNoEsp > 0 then ▷ Hay eliminados del skyline no espacial, región podría ser menor
41:       RECALCULARREGIONBUSQUEDANOESPACIAL(skylineNoEspacial, regionBusqueda)
42:     end if
43:   end if
44:   for d  $\in$  disponibles do                  ▷ Actualiza con los nuevos disponibles el skyline
45:     if dNoEsp > 0 then AGREGARCANDIDATONOESP(candidatosNoEspNvos, d, agrCandidatosNoEsp)
46:     if ESTAREGIONBUSQUEDA(regionBusqueda, d) then
47:       AGREGARCANDIDATO(candidatosNvos, skyline, skylineCHdin, skylineCHest, d)
48:       REVISARDOMINANCIA_SKYLINE_SOLO_ESPACIAL(skyline, d)
49:     end if
50:   end for
51:   if elimSkylineNoEsp = 0 and agrCandidatosNoEsp > 0 then          ▷ Caso 1
52:     AGREGARDESCARTADOS_SKYLINE_NO_ESP(skylineNoEsp, candidatosNoEspNvos)
53:     REALIZARSFSSKYLINE(skylineNoEsp, candidatosNoEspNvos)
54:     ADDALL(candidatosNoEsp, candidatosNoEspNvos)
55:   else if elimSkylineNoEsp > 0 and agrCandidatosNoEsp > 0 then          ▷ Caso 2
56:     ADDALL(candidatosNoEsp, candidatosNoEspNvos)
57:     Integer i  $\leftarrow$  OBTENERINDICEREVISION(candidatosNoEsp, candidatosNoEspNvos)
58:     REALIZARSFSSKYLINEINICIO(skylineNoEsp, candidatosNoEsp, i)
59:   else if elimSkylineNoEsp > 0 and agrCandidatosNoEsp = 0 then          ▷ Caso 3
60:     Integer i  $\leftarrow$  OBTENERINDICEREVISION(candidatosNoEsp)
61:     REALIZARSFSSKYLINEINICIO(skylineNoEsp, candidatosNoEsp, i)
62:   end if
63:   if elimSkyline > 0 or elimSkylineNoEsp > 0 then ▷ Si la region cambió, se buscan más candidatos
64:     BUSCARNVOSCANDIDATOS(regionBusqueda, regionPrev, candidatos)
65:     DEPURARCANDIDATOS(candidatos, regionBusqueda)           ▷ Elimina candidatos  $\notin$  regionBusqueda
66:     CALCULARSKYLINE(skyline, candidatos)                     ▷ Actualiza el skyline, excluye candidatosNvos
67:     ADDALL(candidatos, candidatosNvos)
68:   else
69:     if EXISTENNVOS_SKYLINE(skyline, skylineNoEsp) then REASIGNARSKYLINE(skyline, skylineNoEsp)
70:     ADDALL(candidatos, candidatosNvos, candidatosTmp)
71:   end if
72:   ADDALL(skyline, skylineNoEsp)
73:   return skyline                                     ▷ Skyline actualizado
74: end procedure

```

3.4.3.3 Modo actualización de desplazamiento

El procedimiento CALCULARSOLUCIONDESPLAZAMIENTO (Algoritmo 12, p. 81, Líneas 75-101) es ejecutado para iniciar el *modo actualización de desplazamiento*, al cual se le pasa la nueva posición del punto de referencia móvil. Se almacena la región de búsqueda anterior en la variable `regionPrev` (Línea 76). Al igual que el algoritmo *VCS*², almacena el *convex hull* anterior en la variable `chPrev` (Línea 77), obtiene el nuevo y actualiza la nueva posición del punto de referencia (Línea 78). Asimismo, se genera el sector para el cual el desplazamiento no afectó el *skyline*, tomando el *convex hull* previo y actual (Línea 79) e identifica el patrón de cambio ocurrido (Línea 80).

Para el caso 1 (Línea 81), debido a que continua siendo el mismo polígono `ch`, no es necesario actualizar la distancia al punto móvil en cada objeto, ya que esa dimensión no es tomada en cuenta en la dominancia por no ser vértice de `ch`. Se devuelve el mismo conjunto `skyline`.

Si existen dimensiones no espaciales (Línea 82), se actualiza la dimensión correspondiente al punto móvil de los objetos en la lista `skylineNoEsp` para generar la nueva región.

Para el caso 6 (Líneas 83-86), se actualiza la dimensión correspondiente al punto móvil de los objetos en `skyline` y en caso de no existir dimensiones no espaciales se genera también la nueva región y la almacena en `regionBusqueda` (Línea 84). Además, actualiza el *skyline* mediante `ELIMINAROBJNOSKYLINE` (Línea 85) y `AGREGAROBJAHORASKYLINE` (Línea 86), desecharlo los objetos que ya no deberían estar y añadiendo al *skyline* los objetos que deberían estar, respectivamente.

Para los casos 2 y 4 (Líneas 87-89), invoca al procedimiento `REASIGNARSKYLINEMOV` (Línea 88) que itera la lista `skyline` e identifica por cada elemento si pertenece a otra porción del *skyline*. Es decir, determina si ahora el objeto está dentro del polígono `ch` para reasignarlo. Posteriormente, se ejecuta la función `AGREGAROBJAHORASKYLINE` para actualizar el conjunto *skyline* (Línea 89).

Para el caso 3 y 5 (Líneas 90-98), actualiza las distancias de los elementos del *skyline*, sólo la correspondiente al punto móvil, y si únicamente se tienen dimensiones espaciales, genera la nueva región, `regionBusqueda` (Línea 91). Se eliminan aquellos objetos que ya no deben estar en el *skyline* con `ELIMINAROBJNO_SKYLINE` (Línea 92) y para el caso 3 (Líneas 93-94), debido a que no se actualizaron las distancias de los candidatos, debe efectuarse el procedimiento `ACTUALIZARDISTANCIAPTODIN` (Línea 94) y así dejar los datos actualizados para la siguiente ejecución del algoritmo. Para el caso 5 (Líneas 95-97), adicionalmente se deben revisar los objetos a agregar al *skyline* con `AGREGAROBJAHORASKYLINE` (Línea 96).

Finalmente, se insertan a la lista `skyline` aquellos objetos en `skylineNoEsp` que no estaban y se retorna el conjunto respuesta (Líneas 99-100).

Es importante destacar que la implementación de este algoritmo se realizó minimizando las búsquedas en las listas o índices utilizados, por lo que se emplean etiquetas para marcar que el objeto tiene cierta propiedad y en lugar de efectuar búsquedas en las estructuras, se revisa esa etiqueta. Ademas para evitar redundancia, en esas estructuras sólo se almacenan referencias al mismo objeto. Se utilizan etiquetas para:

- ★ Identificar a cuáles porciones del *skyline* pertenece el objeto: no espacial, fuera del *convex hull*, dentro del *convex hull* estático o dentro del *convex hull* dinámico. Dependiendo de la clasificación, se hacen sólo las verificaciones de dominancia correspondientes.
- ★ Indicar que ya el objeto fue revisado en algún paso del algoritmo y que no es necesario tomarlo en cuenta para otras evaluaciones.
- ★ Saber si el elemento pertenece a la lista de candidatos.
- ★ Forzar la revisión de un subconjunto de los objetos en alguna verificación de dominancia del *skyline*.

Algoritmo 12 CD2S (Parte 3)

▷ Modo actualización de desplazamiento

```

75: procedure CALCULARSOLUCIONDESPLAZAMIENTO(nuevaPosPtoRef)
76:   Rectangle regionPrev  $\leftarrow$  regionBusqueda
77:   Polygon chPrev  $\leftarrow$  ch
78:   ch  $\leftarrow$  ACTUALIZARCONVEXHULL(ptosRef, nuevaPosPtoRef)
79:   Polygon sectorSinCambio  $\leftarrow$  GENERARSECTORSCINCAMBIO(ch, chPrev)
80:   caso  $\leftarrow$  IDENTIFICARCASODESPLAZAMIENTO(ch, chPrev)
```

▷ Caso 1. No se realizan cambios en el skyline

```

81:   if caso = 1 then return skyline
82:   if dNoEsp > 0 then regionBusqueda  $\leftarrow$  OBTENERNUEVARREGIONBUSQUEDA(skylineNoEsp)
83:   if caso = 6 then                                ▷ Caso 6. Agrega y elimina objetos del skyline
84:     regionBusqueda  $\leftarrow$  ACTUALIZARDISTANCIAPTODINREGION(skyline, attObj, nuevaPosPtoRef)
85:     ELIMINAROBJNOSKYLINE(skyline, skylineCHdin, ch, regionBusqueda)
86:     AGREGAROBJAHORASKYLINE(skyline, ch, regionBusqueda, regionPrev, candidatos)
```

▷ Casos 2 y 4. Sólo agrega objetos al skyline

```

87:   else if caso = 2 or caso = 4 then          ▷ Casos 2 y 4. Sólo agrega objetos al skyline
88:     regionBusqueda  $\leftarrow$  REASIGNARSKYLINEMOV(skyline, ch, attObj, nuevaPosPtoRef)
89:     AGREGAROBJAHORASKYLINE(skyline, ch, regionBusqueda, regionPrev, candidatos, sectorSinCambio)
```

▷ Casos 3 y 5

```

90:   else
91:     regionBusqueda  $\leftarrow$  ACTUALIZARDISTANCIAPTODINREGION(skyline, attObj, nuevaPosPtoRef)
92:     ELIMINAROBJNOSKYLINE(skyline, ch, regionBusqueda, sectorSinCambio)
93:     if caso == 3 then                          ▷ Caso 3. Sólo eliminó objetos del skyline
94:       ACTUALIZARDISTANCIAPTODIN(candidatos, attObj, nuevaPosPtoRef)
95:     else                                         ▷ Caso 5. Agrega y elimina objetos del skyline
96:       AGREGAROBJAHORASKYLINE(skyline, ch, regionBusqueda, regionPrev, candidatos, sectorSinCambio)
97:     end if
98:   end if
99:   ADDALL(skyline, skylineNoEsp)
100:  return skyline                                ▷ Conjunto skyline actualizado
101: end procedure
```

▷ Extrae de skylineChdin objetos que ya no están en ch y del skyline los que ya no pertenecen a él

```

102: procedure ELIMINAROBJNOSKYLINE(skyline, ch, regionBusqueda, sectorSinCambio)
103:   EVALUARELIMINAROBJCHDINAMICO(skyline, skylineCHdin, ch, regionBusqueda, sectorSinCambio)
104:   EVALUARELIMINAROBJNOSKYLINE(skyline, skylineCHdin, ch, regionBusqueda, sectorSinCambio)
105: end procedure
```

▷ Agrega y busca los nuevos miembros del skyline

```

106: procedure AGREGAROBJAHORASKYLINE(skyline, regionBusqueda, regionPrev, candidatos)
107:   Dado c no explorado: si c  $\notin$  regionBusqueda, lo elimina de candidatos; si c  $\in$  ch, lo agrega a skylineCHdin
108:   REVISARCANDIDATOS(skyline, skylineCHdin, regionBusqueda, candidatos, ch)
109:   BUSCARNVOSCANDIDATOS(regionBusqueda, regionPrev, candidatos)
110:  CALCULARSKYLINE(skyline, candidatos, sectorSinCambio)                                ▷ Actualiza el skyline
111: end procedure
```

▷ Busca candidatos fuera de la región común ya revisada y dentro de la nueva region

```

112: procedure BUSCARNVOSCANDIDATOS(regionBusqueda, regionPrev, candidatos)
113:   Rectangle regionBusquedaTmp  $\leftarrow$  OBTENERINTERSECCION(regionPrev, regionBusqueda)
114:   if not EQUALS(regionBusquedaTmp, regionBusqueda) then
115:     REVISARALREDEDORESREGION(regionBusquedaTmp, regionBusqueda, plano, candidatos)
116:   end if
117: end procedure
```

CAPÍTULO IV

DISEÑO DEL ESTUDIO EXPERIMENTAL

Debido a que no se cuenta con ambientes reales de pruebas, se hicieron estudios basados en simulaciones con el objetivo de probar el desempeño de los algoritmos.

Los algoritmos implementados antes de ser comparados en el estudio experimental, fueron probados tomando desde 100 a 15.000 objetos de entrada y revisados minuciosamente para garantizar el buen funcionamiento de los mismos, en cada una de las ramas de ejecución. En caso de existir incronguencias entre los objetos sugeridos como *skyline* por un algoritmo y por otro, se estudió la traza de la ejecución de los algoritmos para identificar los errores y solventarlos. Este proceso tuvo una duración aproximada de 10 semanas.

Posteriormente, se procedió al estudio experimental empleando distintas configuraciones iniciales, por lo que se elaboraron siete configuraciones en total. Una **configuración** es el conjunto de valores de los siguientes parámetros o condiciones:

1. **#Objetos**: es la cantidad de objetos espaciales o localidades contenidas en el plano. Se supone que cada localidad en el plano es de $1m \times 1m$. Los experimentos se realizaron con 12.800 a 100.000 objetos.
2. **Ocupados**: es el porcentaje de localidades que inicialmente no están disponibles. En los experimentos realizados se fijó en un 20 %, para no dejar todos los objetos como disponibles, así el área dada por *skyline* puede ser un poco mayor a que si estuviesen todas las localidades disponibles.
3. **CHMax**: es el porcentaje del área de la región máxima del plano donde pueden estar situados los puntos de referencia que en caso de los algoritmos VC2S+ y CD2S gene-

rarán el *convex hull* inicial. Por ejemplo, si el plano tiene de largo 10m y ancho 50m, el área total sería de 500m². Si se define $CHMax = 50\%$, entonces el área cubierta por el *convex hull* formado por los puntos de referencia debe ser menor o igual a 250m². En los experimentos realizados varía este parámetro de 10% a 100%.

4. **#ptosRef:** es la cantidad de puntos de referencia o dimensiones espaciales, cuyas posiciones son escogidas de forma aleatoria. Los experimentos realizados varían de 5 a 18 puntos de referencia.
5. **#dimNoEsp:** es la cantidad de dimensiones no espaciales que poseen los objetos, los valores de estos atributos son escogidos de forma aleatoria. Las dimensiones no espaciales son las que permitirían simular alguna característica no espacial de los objetos, por ejemplo: el tiempo de salida desde esa localidad, el costo por hora de permanecer en esa localidad o cualquier otro atributo que se desee minimizar. Los experimentos realizados varían de 0 a 3 dimensiones no espaciales.

A partir de estas configuraciones se construyeron aleatoriamente 248 casos de prueba, donde un **caso de prueba** es una instancia de los archivos mencionados en el Capítulo III, Sección 3.1: **aConf**, **aPtosRef** y **aDimNoEsp** (este último es llenado si existen dimensiones no espaciales), los cuales conforman la entrada de los algoritmos para la fase de inicialización en los distintos estudios. Para mayor información referirse al Apéndice A.

Con el fin de minimizar errores de generalización y tener una muestra heterogénea en los resultados obtenidos de cada experimento, con cada caso de prueba, se hicieron al menos 500 simulaciones divididas en 10 lotes. Un **lote** es un conjunto de 50 simulaciones seguidas, es decir, partiendo de un caso de prueba con el que se inician las ejecuciones de los algoritmos en el *modo inicial*, se ejecutan 49 simulaciones restantes (según los parámetros de transformación dados para cada simulación) y una vez terminado ese lote, se repite el procedimiento hasta tener 10 lotes. Existen casos de prueba con los que se realizaron tres grupos de 500 simulaciones, pero con distintos parámetros de simulación, ya que se utilizaron en más de un estudio, estos son los casos de prueba de la configuración 2 empleados en los experimentos II, III y IV.

En total se realizaron 436.000 simulaciones que se convirtieron en 436.000 ejecuciones para cada uno de los algoritmos: IB2S, VC2S+ y CD2S; y 420.000 ejecuciones para el algoritmo 2B2S, esta diferencia se debe a que en el experimento VI se excluyó este último algoritmo por ser más ineficiente.

En las secciones 4.1, 4.2, 4.3, 4.4 y 4.5 se explican las diferentes configuraciones de los datos de entrada, los casos de prueba generados para estas configuraciones, los parámetros empleados en las simulaciones, la arquitectura utilizada y las métricas consideradas.

4.1 Configuraciones

Las configuraciones y casos de prueba de los experimentos están en la Tabla 4.1. Donde:

- * **Exp:** corresponde al experimento para el cual fue empleada la configuración.
- * **Conf:** indica la configuración para la cual fueron generados los casos de prueba.
- * **Plano:** es el ambiente de entrada donde se sitúan las localidades y los puntos de referencia. Puede ser modificado por los parámetros: #Objetos, Ocupados y CHMax.
- * **# Casos de prueba:** indica la cantidad casos de prueba construidos para los valores de **Puntos de referencia**, **Dimensiones no espaciales** especificados y el **Total** de casos de prueba para esa configuración.

Exp	Conf	Plano			# Casos de prueba								Total
		#Objetos	Ocupados	CHMax	Puntos de referencia				Dimensiones no espaciales				
					5	7	10	12	14	16	18	0	1
I	1	12.800	20%	100%	16	16	16	0	0	0	0	12	12
II, III, IV	2	12.800	20%	10%	16	16	16	0	0	0	0	12	12
II	3	12.800	20%	50%	16	16	16	0	0	0	0	12	12
V	4	51.200	20%	10%	8	8	8	0	0	0	0	6	6
V	5	51.200	20%	25%	8	8	8	0	0	0	0	6	6
V	6	51.200	20%	50%	8	8	8	0	0	0	0	6	6
VI	7	100.000	20%	10%	0	0	0	8	8	8	8	16	16
Total de casos de prueba					72	72	72	8	8	8	8	70	70
												54	54
													248

Tabla 4.1: Casos generados para la inicialización de los algoritmos

4.2 Clasificación de los casos de prueba

Los casos de prueba de las configuraciones antes descritas se clasifican en 16 combinaciones distintas de los parámetros `#ptoRef - #dimNoEsp`. Se construyó más de un caso de prueba para la misma combinación con el objetivo de tener una muestra más homogénea. El detalle de las combinaciones generadas puede verse en la Tabla 4.2. Donde:

- * **Combinación:** corresponde al identificador del par `#ptoRef - #dimNoEsp` empleado en el caso de prueba.
- * **Valores:** indica el valor de los parámetros: `#ptoRef` y `#dimNoEsp`.
- * **# Casos de prueba:** indica la cantidad casos de prueba construidos para los valores de Configuración dados y el Total de casos de prueba para esa combinación.

Combinación	Valores #ptoRef - #dimNoEsp	# Casos de prueba							Total
		1	2	3	4	5	6	7	
1,2,3,4	5-0, 5-1, 5-2, 5-3	4×4	4×4	4×4	2×4	2×4	2×4	0	18×4
5,6,7,8	7-0,7-1,7-2,7-3	4×4	4×4	4×4	2×4	2×4	2×4	0	18×4
9,10,11,12	10-0, 10-1, 10-2, 10-3	4×4	4×4	4×4	2×4	2×4	2×4	0	18×4
13,14,15,16	12-0,14-0,16-0,18-0	0	0	0	0	0	0	4×4	4×4
17,18,19,20	12-1,14-1,16-1,18-1	0	0	0	0	0	0	4×4	4×4
Total de casos de prueba		48	48	48	24	24	24	32	248

Tabla 4.2: Clasificación de los casos de prueba

4.3 Simulaciones

Se desarrolló un simulador que genera aleatoriamente, los cambios del ambiente en cada iteración para así realizar una transformación del mismo partiendo del ambiente modificado en la iteración anterior. Inicialmente, se toma el ambiente dado por el caso de prueba.

Cada simulación producida es común para los algoritmos, es decir, una misma simulación es tomada como entrada para cada algoritmo utilizado en el estudio con el objetivo de compararlos bajo las mismas condiciones. Una vez que los algoritmos hayan terminado su ejecución y retorna el conjunto *skyline* completo, se genera la siguiente simulación. Con cada simulación se genera una ejecución del algoritmo en el modo de ejecución correspondiente.

Los parámetros utilizados para generar una simulación, se muestran a continuación:

1. **Modo:** establece el modo en que son ejecutados los algoritmos según el simulador, puede ser: *modo actualización de estatus* en todas las simulaciones, *modo actualización de desplazamiento* en todas las simulaciones, o combinado, donde aleatoriamente el simulador escoge en cada simulación el cambio a realizar. El *modo inicial* siempre se ejecutará en la primera simulación para inicializar los algoritmos con el caso de prueba.
2. **%Cambio:** es el parámetro utilizado para establecer el porcentaje de localidades que cambian de estatus en el plano en cada simulación, los objetos que cambian de estatus son escogidos por el simulador de forma aleatoria hasta cubrir el valor dado. Este parámetro puede ser un valor fijo o un valor aleatorio que cambie en cada simulación.
3. **DistMax:** limita la distancia máxima a la que el punto de referencia móvil se puede desplazar en una simulación, escogiendo de forma aleatoria la dirección del desplazamiento: hacia arriba, abajo, izquierda o derecha, según las dimensiones del plano.

En la metodología de cada experimento se especifican los valores de estos parámetros, aunque pueden observarse también en el Apéndice F.

4.4 Arquitectura utilizada

Las simulaciones se realizaron según cada experimento en las siguientes plataformas:

- * Experimento I: en un equipo Dell XPS XPS720 con procesador Intel(R) Core(TM)2 Quad CPU Q6600 @ 2.40GHz, 4MB Cache, 4GB RAM y 1TB de disco duro. Utilizando la versión de Java 1.6.0_20 32-Bit con los siguientes parámetros para el JVM:
-Xss2m -Xms32m -Xmx512m -XX:PermSize=32m -XX:MaxPermSize=200m.
- * Experimento II, III, IV y V: en un equipo Dell Inspiron 5520 con procesador Intel(R) Core(TM) i5-3210M CPU @ 2.50GHz, 3MB Cache, 6GB RAM y 1TB de disco duro. Utilizando la versión de Java 1.6.0_33 64-Bit con los siguientes parámetros para el JVM: -Xss2m -Xms32m -Xmx768m -XX:PermSize=32m -XX:MaxPermSize=384m.

- * Experimento IV: en un equipo Dell Inspiron 5520 con procesador Intel(R) Core(TM) i5-3210M CPU @ 2.50GHz, 3MB Cache, 6GB RAM y 1TB de disco duro. Utilizando la versión de Java 1.6.0_24 64-Bit con los siguientes parámetros para el JVM: -Xverify:none -Xss5m -Xms1g -Xmx4g -XX:PermSize=128m -XX:MaxPermSize=1024m.

4.5 Métricas, indicadores y medidas

En los estudios realizados se emplearon diversas métricas, indicadores y medidas para analizar los resultados agrupándolos por lotes, combinaciones o variantes del experimento. Una **variante** es un conjunto de simulaciones realizadas, que comparten los mismos parámetros, para una configuración en particular.

1. **Tiempos:** todas las mediciones se realizaron en milisegundos (ms).

Tiempo por lote: período en que el algoritmo ejecuta 50 simulaciones seguidas.

Tiempo promedio por lote: corresponde al **tiempo por lote** dividido entre el número de lotes de la combinación dada. Es igual a: $t_{Ini} + t_{Est} + t_{Mov}$.

t_{Ini} : tiempo promedio por lote empleado en el *modo inicial* para una combinación dada, se calcula como la sumatoria del tiempo de ejecución del algoritmo en el *modo inicial* dividido entre el número de lotes de esa combinación. **t_{Est} :** similar al t_{Ini} con el *modo actualización de estatus*. **t_{Mov} :** similar al t_{Ini} con el *modo actualización de desplazamiento*.

2. **Transformaciones realizadas por el simulador**

#vecesEst: número promedio de simulaciones realizadas en un lote con el *modo actualización de estatus* para una variante dada.

#vecesMov: similar al **#VecesEst** con el *modo actualización de desplazamiento*.

Desplazamiento: distancia promedio recorrida por el punto de referencia móvil en un lote para una variante dada.

#Cambios: promedio de cambios de disponibilidad realizados en un lote para una variante dada.

3. Procesamiento del *skyline*

#Objetos comparados: número promedio de la cantidad acumulada de objetos comparados para la verificación de dominancia del algoritmo en un lote para una combinación dada.

#Comparaciones de dominancia: número promedio de la cantidad acumulada de comparaciones realizadas por el algoritmo en un lote para una combinación dada, con el fin de: descartar, agregar o eliminar el candidato al *skyline* o el objeto de la ventana.

#Candidatos al *skyline*: cardinalidad acumulada de los candidatos al *skyline* en un lote para una combinación o una variante dada. Para el algoritmo VC2S+ se utiliza el conjunto de visitados y para el resto de los algoritmos, se toma el de candidatos. Para el algoritmo CD2S, los objetos que pertenecen al sector sin cambio están en ese conjunto aún cuando no se consideren.

4. Composición del *skyline*

Skyline: promedio de la cardinalidad del *skyline* acumulada en un lote para una combinación o una variante dada. **s:** promedio de la cardinalidad acumulada del conjunto *skyline* que posee objetos fuera del *convex hull* en un lote para una combinación o una variante dada. **sDin:** promedio de la cardinalidad acumulada del conjunto *skyline* dinámico en un lote para una combinación o una variante dada. **sEst:** promedio de la cardinalidad acumulada del conjunto *skyline* estático en un lote para una combinación o una variante dada. **sNE:** promedio de la cardinalidad acumulada del conjunto *skyline* no espacial en un lote para una combinación o una variante dada.

5. Cambios en el *skyline*

+s: promedio de las veces que el algoritmo agrega un elemento al *skyline* espacial en un lote para una combinación dada. **-s:** similar a +s, pero para la eliminación. **+sNE:** similar a +s, pero para el *skyline* no espacial. **-sNE:** similar a +sNE, pero para la eliminación.

6. Casos de desplazamiento

1-5: cantidad de veces promedio que el algoritmo identificó cada patrón de cambio en un lote para una variante dada. **Otros:** similar, indica las veces que CD2S consiguió otro patrón distinto. **Reiniciar:** similar a **Otros**, indica las veces que el algoritmo VC2S+ tuvo que calcular el *skyline* desde el inicio.

7. Distribución del tiempo de ejecución del algoritmo VC2S+

tCreacionDelaunay: tiempo promedio por lote empleado en la creación del grafo *Delaunay* para una variante dada. **tCalculoDistanciaVC:** tiempo promedio por lote empleado en el cálculo de distancia mínima de cada celda de *Voronoi* a las dimensiones espaciales para una variante dada. **tRestante:** tiempo promedio por lote menos los valores de **tCreacionDelaunay** y **tCalculoDistanciaVC**

8. Medidas de tendencia central

Media (μ): tiempo promedio de ejecución del algoritmo en un lote para una variante o experimento dado. **Mediana (M_e):** punto medio del **tiempo por lote** del algoritmo en los lotes de una variante dada. Los datos menores o iguales que la mediana representan el 50 % de la muestra y los mayores el otro 50 %. **Moda (M_o):** **tiempo por lote** que más se repite para una variante dada. **Frecuencia absoluta modal (f_{M_o}):** cantidad de veces que se repite la moda.

9. Medidas de dispersión

Desviación estándar (σ): medida del grado de dispersión de los datos con respecto a la **media** para una variante o experimento dado.

10. Indicadores

Razón $\frac{\mu_i}{\mu_4}$: proporción de la **media** del algoritmo CD2S con respecto a la **media** del algoritmo i para una variante dada. Indica el número de veces que la **media** de CD2S es menor que la de ese algoritmo. Si $\frac{\mu_i}{\mu_4} < 1$, indica que la **media** de CD2S es mayor a la de ese algoritmo. **vecesTMin:** número de veces en que el **tiempo por lote** de un algoritmo es menor que el resto para una variante o combinación dada. **vecesTMax:** similar a **vecesTMin**, pero para el caso en que es mayor. **Mejor tiempo:** menor **tiempo por lote** registrado entre todos los algoritmos para una combinación dada. **Peor tiempo:** mayor **tiempo por lote** registrado.

CAPÍTULO V

ESTUDIO EXPERIMENTAL

Con la finalidad de observar el desempeño de los diferentes algoritmos, se realizaron seis experimentos basados en simulaciones. El Experimento I estudia el comportamiento general de los algoritmos; el Experimento II indica cómo se ve afectado ese comportamiento al establecer límites en el área del *convex hull* inicial; los Experimentos III y IV analizan el desempeño de los algoritmos utilizando sólo el *modo actualización de desplazamiento* y el *modo actualización de estatus*, respectivamente; el Experimento V evalúa los algoritmos en presencia de una cardinalidad media de objetos y el Experimento IV realiza un análisis de los algoritmos bajo una cardinalidad alta de objetos para calcular el *skyline* con sólo dimensiones espaciales y con una dimensión no espacial. En las secciones 5.1, 5.2, 5.3, 5.4, 5.5 y 5.6 se realiza el análisis de estos experimentos.

5.1 Experimento I

El objetivo de este experimento es estudiar el comportamiento general de los algoritmos implementados respecto a la variación del número de puntos de referencia y dimensiones no espaciales. Un crecimiento en las dimensiones, produce una mayor cantidad de objetos incomparables y de comparaciones de dominancia necesarias para el *skyline* [88].

Se tienen las siguientes hipótesis:

- H1** El algoritmo CD2S requiere menos tiempo respecto a los otros algoritmos implementados para obtener el *skyline*.
- H2** La cardinalidad del *skyline* aumenta a medida que incrementa la cantidad de puntos de referencia y las dimensiones no espaciales.

5.1.1 Metodología

Este experimento se realizó mediante los casos de prueba de la configuración 1 (Tabla 4.1, Capítulo IV), con cada variante se generaron 10 lotes para cada caso de prueba, en total 480 lotes, a través de las siguientes simulaciones:

- * **Variante 1:** Conf=1, Modo=combinado, %Cambio=aleatorio y DistMax=10.
- * **Variante 2:** Conf=1, Modo=combinado, %Cambio=aleatorio y DistMax=20.

5.1.2 Resultados

En la Tabla 5.1 se muestran en las primeras 4 filas, la **media** μ_i de cada algoritmo en milisegundos (ms) y la razón $\frac{\mu_i}{\mu_4}$ de cada uno en comparación con el algoritmo CD2S para cada variante ($DistMax=10, 20$) y en general para el experimento, donde se agrega la **desviación estándar** σ_i . En las siguientes 4 filas, se colocan las métricas relacionadas a la transformaciones realizadas por el simulador.

Algoritmo	Variante 1 DistMax=10		Variante 2 DistMax=20		General	
	μ_i	$\frac{\mu_i}{\mu_4}$	μ_i	$\frac{\mu_i}{\mu_4}$	μ_i	σ_i
(1) 2B2S	697.579	1,79	691.686	1,75	694.632	627.728
(2) IB2S	639.340	1,64	634.074	1,61	636.707	591.983
(3) VC2S+	745.041	1,91	744.396	1,89	744.718	709.895
(4) CD2S	390.351	1,00	394.318	1,00	392.334	380.064
Desplazamiento	131,50 m		244,00 m		187,75 m	
#Cambios	154.665,00 obj		156.238,50 obj		155.451,75 obj	
#vecesEst	24,33 veces		24,58 veces		24,46 veces	
#vecesMov	24,67 veces		24,50 veces		24,59 veces	

Tabla 5.1: Resultados por lotes de 50 simulaciones

Se puede observar en la Tabla 5.1, cómo el algoritmo CD2S tiene mejor desempeño en cuanto al tiempo empleado para retornar el *skyline* en ambas variantes, siendo 1,61 veces menor ($\frac{\mu_i}{\mu_4}$) que IB2S, 1,75 veces menor que 2B2S y hasta 1,89 veces menor que VC2S+ para $DistMax = 20$. También se puede apreciar, que la menor **media** μ_i y **desviación estándar** σ_i pertenecen al algoritmo CD2S y la mayor al VC2S+. Esto quiere decir que el algoritmo CD2S es el que menos variación tiene en sus tiempos de ejecución, aún cuando los datos tengan una alta desviación porque los resultados son muy distintos entre las

distintas combinaciones (desde la 1 hasta la 12), sin embargo a mayor desviación estándar, más irregular es el comportamiento del algoritmo.

En cuanto a las modificaciones realizadas por el simulador en la Tabla 5.1, el **desplazamiento** en un lote, aumenta de 131,50 metros (variante 1) a 244 metros (variante 2), esto es porque *DistMax* es mayor. El valor de **#Cambios** se mantiene muy similar porque el **%Cambio** de localidades se escogió al azar en cada simulación de ambas variantes. Los valores de **#vecesEst** y **#vecesMov** son parecidos, resultando que el 48 % aproximadamente de las simulaciones corresponden al *modo actualización de estatus*, otro 48 % al *modo actualización de desplazamiento* y un 2 % a la inicialización de los algoritmos en el *modo inicial*.

5.1.2.1 Estudio del tiempo de los algoritmos

En la Figura 5.1 se puede observar con más detalle el tiempo de los algoritmos por combinaciones de los casos de prueba para ambas variantes. En el eje *x* de esta gráfica están las **combinaciones** y en el eje *y* el **tiempo promedio por lote** en ms, las líneas punteadas indican el **mejor tiempo** y **peor tiempo** registrado por algún algoritmo y para algún lote en cada combinación. Se observa que el algoritmo CD2S tiene el menor tiempo de ejecución promedio en todas las combinaciones de ambas variantes, acercándose al mejor tiempo registrado, es por esto que se **cumple H1**.

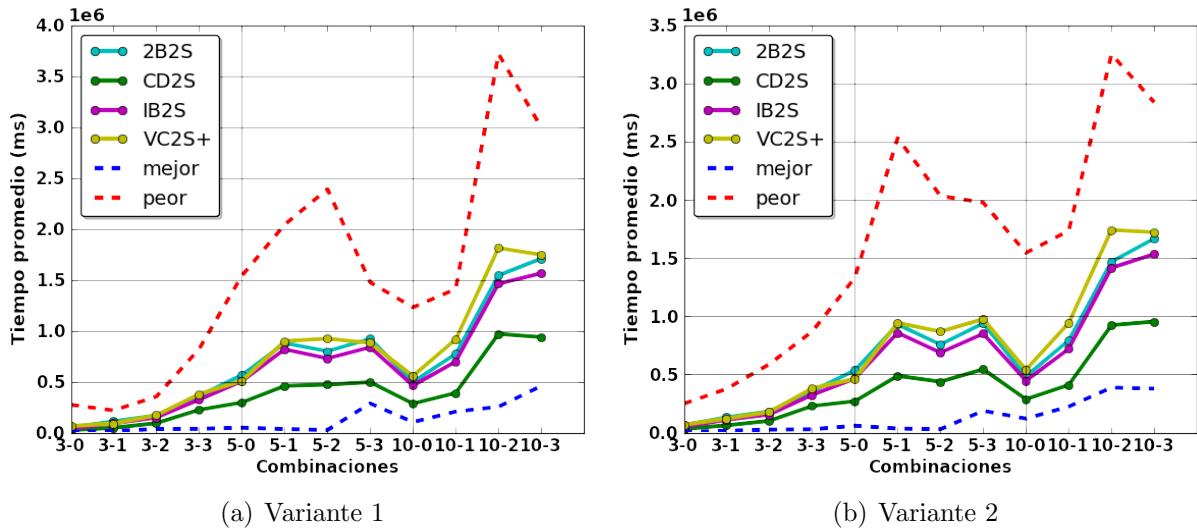
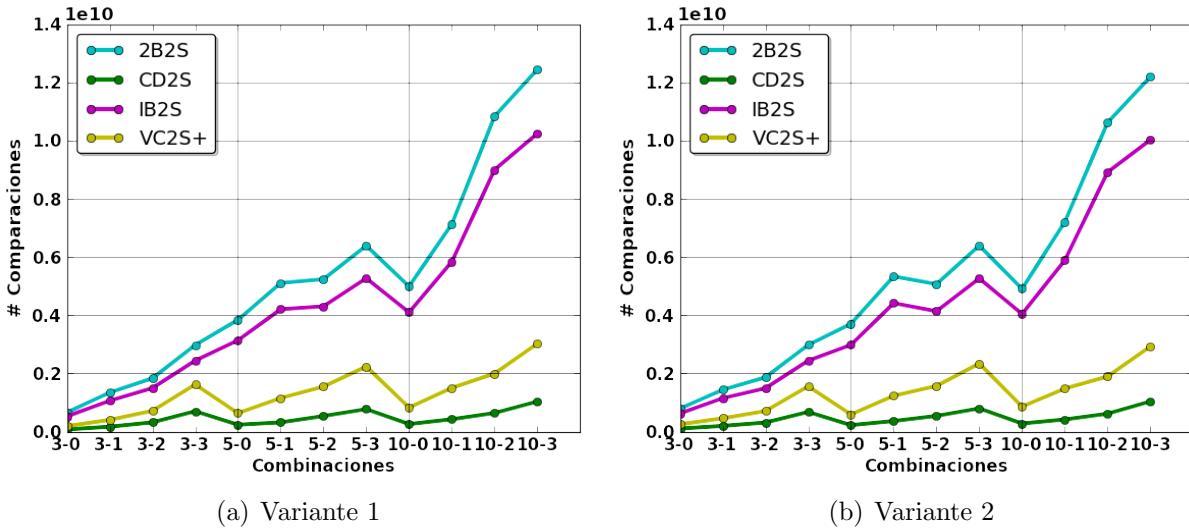


Figura 5.1: Tiempo de ejecución promedio por lote

En general se aprecia que los tiempos crecen a medida que aumentan las dimensiones del *skyline*.

El buen desempeño de CD2S, se debe a que este algoritmo realiza una menor cantidad de comparaciones de dominancia para obtener y actualizar el *skyline*, lo cual puede ser observado en la Figura 5.2. Esta gráfica, para ambas variantes, muestra en el eje *x* las combinaciones y en el eje *y* el #comparaciones. Al igual que la Figura 5.1 se percibe un crecimiento al aumentar las dimensiones.

El algoritmo VC2S+ aún cuando necesita realizar menos comparaciones para obtener el conjunto *skyline* que IB2S y 2B2S, al ser ejecutado en *modo actualización de estatus* debe crear el nuevo grafo *Delaunay*, reiniciar el conjunto *skyline* y calcularlo desde el inicio con el algoritmo *VS*² el cual es mas costoso que el *VCS*². Por esta razón VC2S+ tiene un peor desempeño que los algoritmos básicos IB2S y 2B2S ya que aún cuando no podan el espacio de búsqueda, éstos se ven favorecidos por la estructura matricial para almacenar los estatus de los objetos quedando elementos similares (en dimensiones espaciales) contiguos en la matriz.



(a) Variante 1

(b) Variante 2

Figura 5.2: Comparaciones efectuadas en promedio por lote

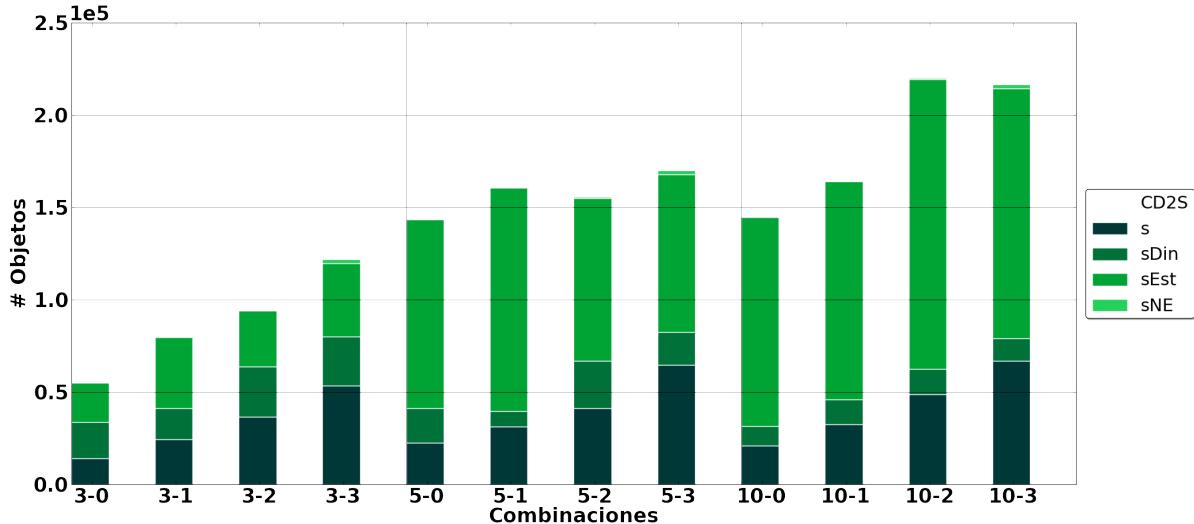
5.1.2.2 Estudio de la cardinalidad del *skyline*

Aún cuando es la misma la cardinalidad de objetos para todas las combinaciones, el incremento en el tiempo de ejecución y en el número de comparaciones, a medida que aumentan las dimensiones (Figuras 5.2 y 5.1), se debe a que el *skyline* crece porque aumenta la cantidad de objetos indistinguibles entre sí, ya que difícilmente existirá un objeto suficientemente bueno en todas las dimensiones.

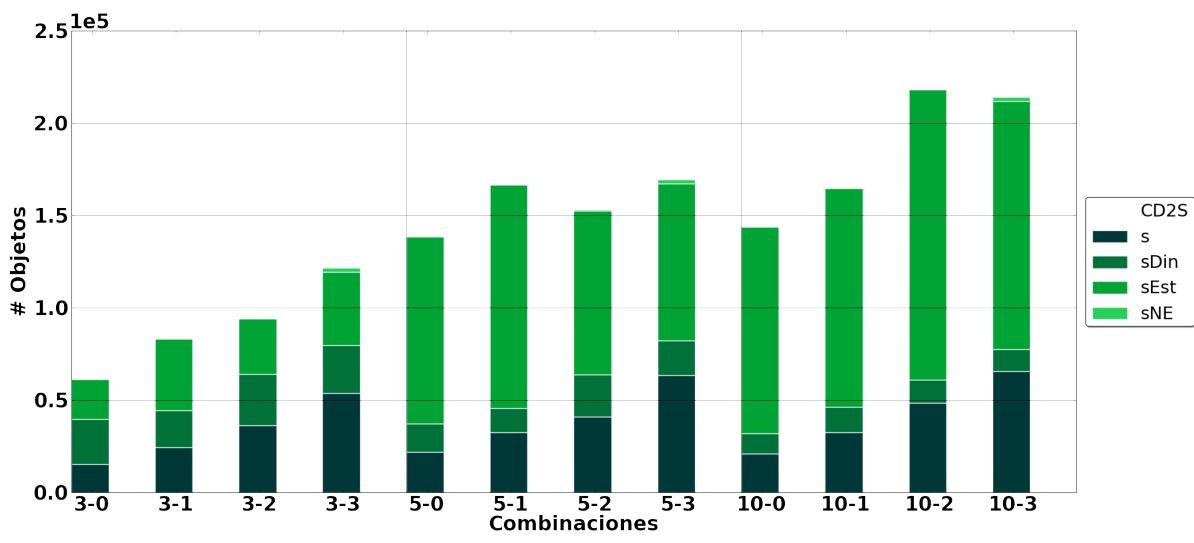
El aumento de la cardinalidad del *skyline* puede verse en la Figura 5.3 que muestra la composición del *skyline*, donde en el eje *x* están las **combinaciones** y en el *y* el **#Objetos**. Para la variante 1, en la combinación 3-0 el tamaño del *skyline* es de 55.101, el cual dividido entre la cantidad de simulaciones de un lote da un promedio de 1.102 objetos por simulación, casi un 8 % de la cantidad de objetos, y en la combinación 10-3 es de 216.541 objetos que corresponde a 4.330 objetos por simulación y representa a casi un 34 % de los objetos, implicando que el *skyline* perdió selectividad por el motivo antes expuesto. Para la variante 2, existe un comportamiento similar, varía la cardinalidad del *skyline* de 61.342 a 214.186. Observando este aumento de cardinalidad en ambas variantes a medida se agregan más dimensiones al *skyline*, se **confirma** la hipótesis **H2**

Adicional al aumento de la cardinalidad del *skyline*, haciendo referencia a la Figura 5.3 para estudiar la composición del mismo, la mayor parte corresponde al *skyline* estático, **sEst** que agrupa a los objetos del *convex hull* que no son afectados por el desplazamiento del punto de referencia, siendo ésta una ventaja para el algoritmo CD2S porque estos elementos no tienen que ser revisados en el *modo actualización de desplazamiento*, sólo se agregaría o eliminaría uno de estos si el objeto cambiara de estatus. Los elementos marcados como **s** conforman la segunda parte considerable del *skyline* y son aquellos que están fuera del *convex hull*, pero que están lo bastante cerca para no ser dominados por otros; esta fracción es menor a la cuarta parte del *skyline* en los casos 3-0, 5-0 y 10-0 y aumenta su participación en el *skyline* a medida que se agregan dimensiones no espaciales porque son estos también, los objetos cuya combinación de sus dimensiones espaciales - no espaciales los hace interesantes. En menor grado, se encuentran los objetos pertenecientes

al *skyline* dinámico, *sDin* y la fracción correspondiente al *skyline* no espacial *sNE* que conforma la parte menos significativa del *skyline*, siendo sólo gráficamente visible en los casos donde la cantidad de dimensiones no espaciales es igual a 3.



(a) Variante 1



(b) Variante 2

Figura 5.3: Composición promedio del *skyline* por lote

El detalle de los resultados de estas simulaciones por combinación puede ser visto en el Apéndice E en las Tablas E.1 y E.2. Así como también, estudios más particulares pueden ser vistos en el Apéndice H.1.

5.2 Experimento II

El objetivo de este experimento es analizar los algoritmos implementados respecto a la variación del tamaño máximo del *convex hull* inicial $CHMax$ ya que se observó en el Experimento I que el *skyline* se compone en gran parte por los objetos contenidos en él.

Se tienen las siguientes hipótesis:

H1 A medida que la región del *convex hull* inicial $CHMax$ aumenta, se incrementan los tiempos de ejecución de los algoritmos.

H2 El tiempo de ejecución del algoritmo CD2S es el menor.

H3 El área del *convex hull* influye directamente en la cardinalidad del *skyline*.

5.2.1 Metodología

Este experimento se realizó mediante los casos de prueba de las configuraciones 2 y 3 (Tabla 4.1, Capítulo IV). Para la configuración 2 ($CHMax = 10\%$) se crearon las variantes 1 y 2 y para la configuración 3 ($CHMax = 50\%$) las variantes 3 y 4, con cada variante se generaron 10 lotes para cada caso de prueba, en total 480 lotes, a través de las siguientes simulaciones:

- * **Variante 1:** Conf=2, Modo=combinado, %Cambio=aleatorio y DistMax=10.
- * **Variante 2:** Conf=2, Modo=combinado, %Cambio=aleatorio y DistMax=20.
- * **Variante 3:** Conf=3, Modo=combinado, %Cambio=aleatorio y DistMax=10.
- * **Variante 4:** Conf=3, Modo=combinado, %Cambio=aleatorio y DistMax=20.

5.2.2 Resultados

En la Tabla 5.2 se encuentran los resultados para las 4 variantes del experimento donde se puede percibir que las **medias** de los tiempos de los algoritmos aumentan considerable entre $CHMax = 10\%$ y $CHMax = 50\%$, así como también la **desviación estándar**.

Algoritmo	CHMax=10 %						CHMax=50 %					
	Variante 1		Variante 2		General		Variante 3		Variante 4		General	
	DistMax=10	$\frac{\mu_i}{\mu_4}$	DistMax=20	$\frac{\mu_i}{\mu_4}$	μ_i	σ_i	DistMax=10	$\frac{\mu_i}{\mu_4}$	DistMax=20	$\frac{\mu_i}{\mu_4}$	μ_i	σ_i
(1) 2B2S	8.646	2,13	14.077	2,24	11.362	13.197	282.517	3,78	304.500	3,53	293.508	214.038
(2) IB2S	7.240	1,78	11.274	1,80	9.257	9.904	212.055	2,83	230.834	2,68	221.444	164.434
(3) VC2S+	9.515	2,34	13.108	2,09	11.312	7.888	137.331	1,84	158.384	1,84	147.858	114.594
(4) CD2S	4.061	1,00	6.275	1,00	5.168	4.799	74.806	1,00	86.151	1,00	80.479	67.626
Desplazamiento	133,67 m		251,42 m		192,55 m		133,83 m		246,00 m		189,92 m	
#Cambios	157.648,75 obj		155.824,67 obj		156.736,71 obj		156.643,33 obj		156.759,33 obj		156.701,33 obj	
#vecesEst	24,67 veces		24,25 veces		24,46 veces		24,42 veces		24,58 veces		24,5 veces	
#vecesMov	24,33 veces		24,75 veces		24,54 veces		24,58 veces		24,50 veces		24,54 veces	

Tabla 5.2: Resultados por lotes de 50 simulaciones

5.2.2.1 Estudio del tiempo de los algoritmos

En la Tabla 5.2, el crecimiento del tiempo promedio de los algoritmos es de, por lo menos, un orden de magnitud al cubrir un área mayor el *convex hull* inicial.

Esto se observa también, en las gráficas de los tiempos promedio por variante de la Figura 5.4 que muestran en el eje *x* las barras de los algoritmos por variante y en el eje *y* el tiempo promedio.

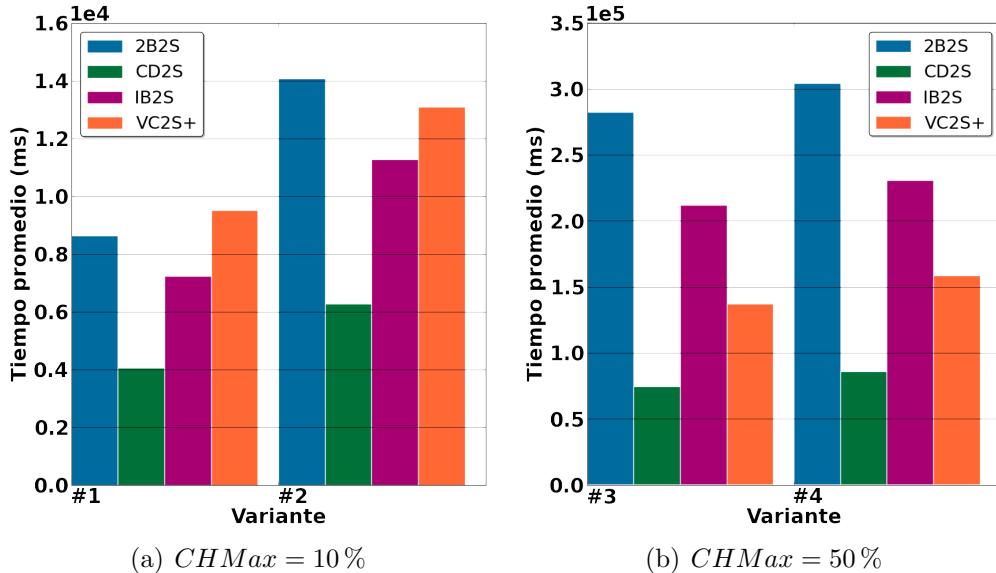
(a) $CHMax = 10 \%$ (b) $CHMax = 50 \%$

Figura 5.4: Tiempo promedio de ejecución por lote

Debido al incremento general del tiempo de los algoritmos en los resultados para $CHMax = 10\%$ comparados con los resultados para $CHMax = 50\%$, se **confirma H1** para este estudio.

Este aumento en la magnitud de los tiempos, se refleja también al crecer de $10m$ a $20m$ la distancia máxima que se puede mover el punto de referencia móvil, y es porque la forma del *convex hull* cambiará más bruscamente.

Observando la Tabla 5.2, los tiempos de los algoritmos 2B2S, IB2Sy VC2S+ son mayores que el tiempo del algoritmo CD2S, siendo la razón $\frac{\mu_i}{\mu_4}$ en los resultados obtenidos, de al menos 1,78 para IB2S y hasta 3,78 para 2B2S. Por lo tanto, debido a que el menor tiempo de ejecución corresponde al algoritmo CD2S en los resultados antes descritos, la hipótesis **H2 se aprueba**.

Esta razón $\frac{\mu_i}{\mu_4}$, es mayor que en el Experimento I, esto se debe a que en el Experimento I no se fijan límites para la posición inicial de los puntos referencia, en el peor caso la tarea de revisar el *convex hull* y sus alrededores podría ser similar a revisar el plano completo para obtener el *skyline*, por lo que aún cuando sea mejor utilizar el algoritmo CD2S no es tan marcada la diferencia en ese experimento.

5.2.2.2 Estudio de la cardinalidad del *skyline*

El promedio de las comparaciones de dominancia, presentes en la Figura 5.5 y en el Apéndice E, que deben realizar los algoritmos aumentan cuantiosamente entre los resultados de la configuración 2 y 3. Esta es la razón por la cual los tiempos de ejecución de los algoritmos crecen también.

Las comparaciones de dominancia mas altas fueron realizadas en la variante 2, estas corresponden al algoritmo 2B2S, donde presenta $17,06 \times 10^8$ comparaciones en la combinación 12 (10-3) y $7,03 \times 10^9$ comparaciones para la misma combinación en la variante 4. La causa real del aumento en el número de comparaciones, es una mayor cardinalidad del *skyline*, que puede ser apreciada en la Figura 5.6. En estas gráficas, en ambas variantes, se observa

que la cardinalidad del *skyline* aumenta al tener más puntos de referencia y dimensiones no espaciales. Así como también, se aprecia que los picos de mayor cardinalidad (combinaciones 4, 8 y 12) son reflejados en las gráficas de la Figura 5.5, con un mayor número de comparaciones para esas mismas combinaciones.

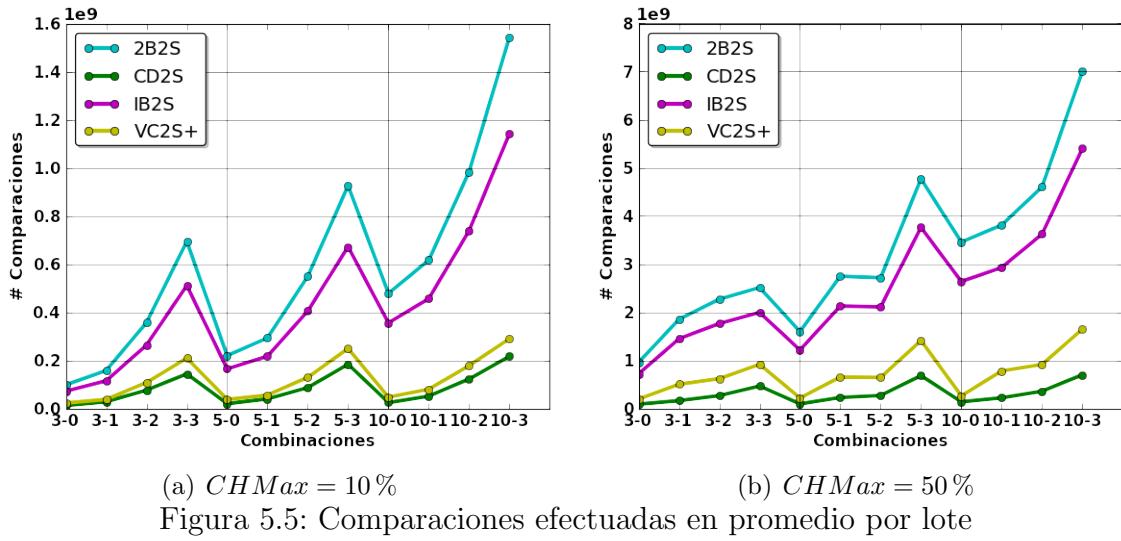
(a) $CHMax = 10\%$ (b) $CHMax = 50\%$

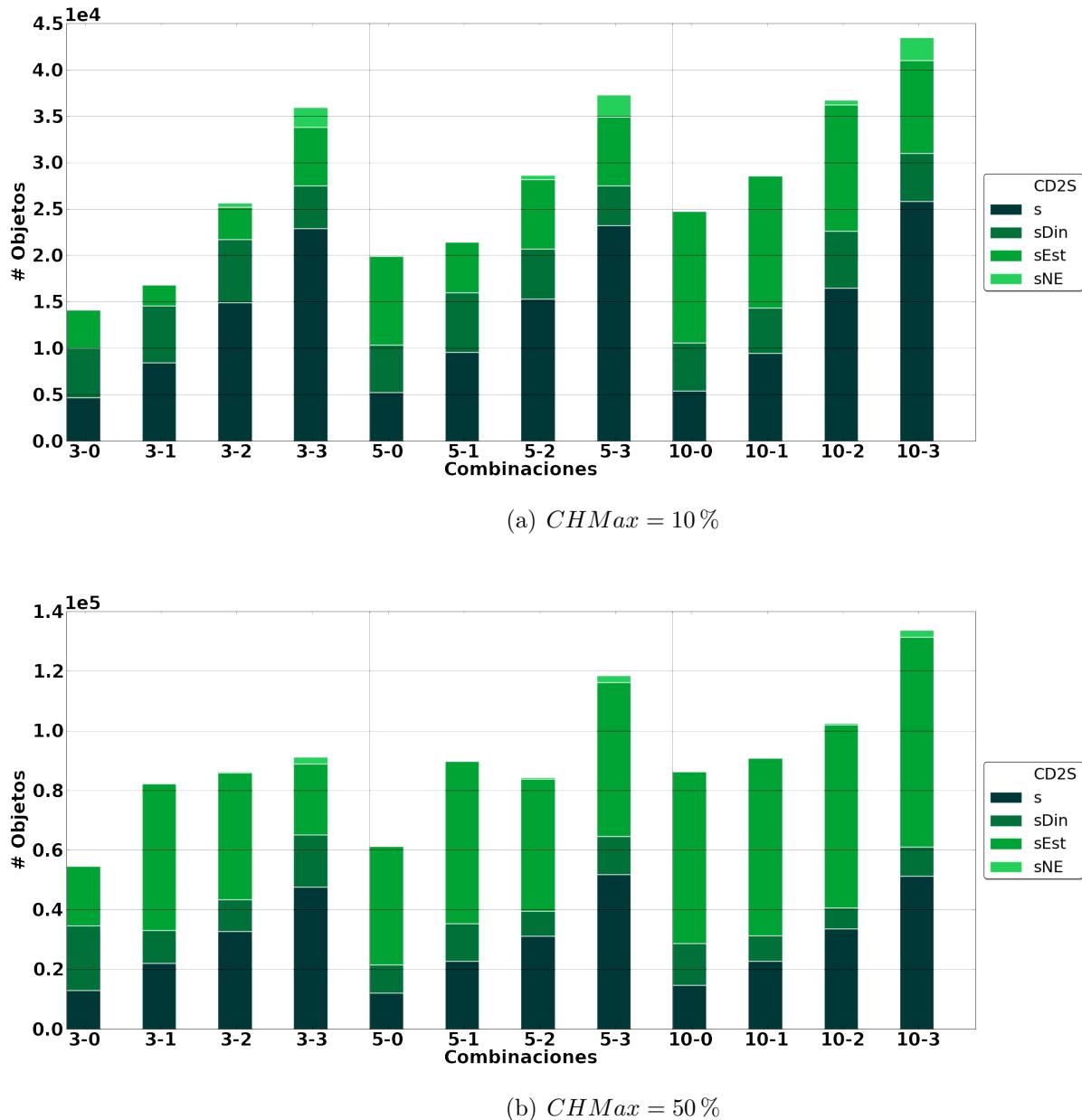
Figura 5.5: Comparaciones efectuadas en promedio por lote

La cardinalidad del *skyline* en la combinación 12 es 46.764 objetos para la variante 2 y 133.404 objetos para la variante 3 y 216,541 para la variante 1 del Experimento I, por lo que se **confirma H3** en este experimento.

Otro resultado esperado, se observa en la composición del *skyline* en la Figura 5.6. Donde se aprecia que al ser pequeño el tamaño inicial del *convex hull* ($CHMax = 10\%$) protagonizan en el *skyline* los elementos que están fuera del *convex hull* y los elementos del *skyline* no espacial tienen una mayor participación en el mismo.

Este resultado, cambia para $CHMax = 50\%$ y $CHMax = 100\%$ donde el *skyline* es formado en mayor parte por los elementos disponibles dentro del *convex hull* (**sDin** y **sEst**).

El detalle de los resultados de estas simulaciones por combinación puede ser visto en el Apéndice E, en las Tablas E.3, E.4, E.15 y E.16. Así como también, estudios más particulares pueden ser vistos en el Apéndice H.2.

Figura 5.6: Composición promedio del *skyline* por lote

5.3 Experimento III

El objetivo de este experimento es analizar el comportamiento de los algoritmos implementados únicamente en el *modo actualización de desplazamiento*, respecto a la variación de la distancia máxima que puede trasladarse el punto de referencia móvil en cada simulación *DistMax*. Los cambios drásticos del *convex hull* de una simulación a otra por el desplazamiento del punto de referencia pueden alterar el *skyline* haciendo que el proceso de actualización requiera más tiempo.

Se tienen las siguientes hipótesis:

- H1** Mientras sea mayor la distancia máxima a recorrer por el punto de referencia móvil *DistMax*, mayor será el tiempo de ejecución.
- H2** El tiempo de ejecución del algoritmo VC2S+ para los casos de 0 dimensiones no espaciales, es el menor entre los demás algoritmos.
- H3** La frecuencia de los patrones de cambio del 1-4 disminuirá y la cantidad de cambios en el *skyline* aumentará a mayor distancia máxima a recorrer.

5.3.1 Metodología

Este experimento se realizó mediante los casos de prueba de la configuración 2 (Tabla 4.1, Capítulo IV), con cada variante se generaron 10 lotes para cada caso de prueba, en total 480 lotes, a través de las siguientes simulaciones:

- * **Variante 1:** Conf=2, Modo=*modo actualización de desplazamiento* y DistMax=5.
- * **Variante 2:** Conf=2, Modo=*modo actualización de desplazamiento* y DistMax=10.
- * **Variante 3:** Conf=2, Modo=*modo actualización de desplazamiento* y DistMax=15.
- * **Variante 4:** Conf=2, Modo=*modo actualización de desplazamiento* y DistMax=20.
- * **Variante 5:** Conf=2, Modo=*modo actualización de desplazamiento* y DistMax=25.

5.3.2 Resultados

En la Tabla 5.3, se observan los resultados de los algoritmos para cada una de las variantes.

Algoritmo	Variante 1 DistMax=5			Variante 2 DistMax=10			Variante 3 DistMax=15			Variante 4 DistMax=20			Variante 5 DistMax=25		
	μ_i	σ_i	$\frac{\mu_i}{\mu_4}$	μ_i	σ_i	$\frac{\mu_i}{\mu_4}$	μ_i	σ_i	$\frac{\mu_i}{\mu_4}$	μ_i	σ_i	$\frac{\mu_i}{\mu_4}$	μ_i	σ_i	$\frac{\mu_i}{\mu_4}$
	(1) 2B2S	21.156	23.779	3,02	35.625	39.695	2,71	51.842	63.793	2,72	62.711	67.328	2,51	80.944	99.319
(2) IB2S	21.181	23.870	3,03	35.648	39.774	2,71	51.848	63.773	2,72	62.758	67.506	2,51	81.012	99.484	2,60
(3) VC2S+	19.936	15.720	2,85	30.080	24.935	2,29	38.775	33.474	2,03	46.899	36.185	1,88	56.920	50.419	1,83
(4) CD2S	6.998	6.772	1,00	13142	12.476	1,00	19.064	18.600	1,00	25.006	21.672	1,00	31.183	29.104	1,00
Desplazamiento	146,83 m			266,67 m			383,83 m			494,75 m			607,67 m		
#vecesMov	49 veces			49 veces			49 veces			49 veces			49 veces		

Tabla 5.3: Resultados por lotes de 50 simulaciones

5.3.2.1 Estudio del tiempo de los algoritmos

Se puede observar en los resultados en la Tabla 5.3 y en la Figura 5.7(f) de tiempos por variantes, que la media crece gradualmente al aumentar el parámetro $DistMax$.

Es importante destacar que, los tiempos de ejecución de los algoritmos para la variante 5 poseen una magnitud mayor al triple, en comparación con los tiempos de los mismos algoritmos para la variante 1, lo cual **comprueba H1** para este conjunto de pruebas, ya que la media del tiempo se ve afectada por los valores del parámetro $DistMax$.

Los tiempos de los algoritmos 2B2S y IB2S son similares para este experimento, es por esto que la linea del IB2S en las gráficas de Tiempo por combinación de las Figuras 5.7(a)-5.7(e), está solapada con la del algoritmo 2B2S, ya que estos algoritmos son idénticos bajo el *modo actualización de desplazamiento*, construyen el *skyline* desde el inicio cada vez que el punto de referencia se mueve.

Se destaca en la Tabla 5.3, el algoritmo CD2S por tener el menor tiempo de ejecución promedio con un valor 2,85 veces menor que el siguiente menor tiempo, correspondiente al algoritmo VC2S+ para la variante 1. Aún cuando, esta diferencia disminuye a 1,83 para la variante 5, el algoritmo CD2S sigue siendo mejor en tiempo de ejecución.

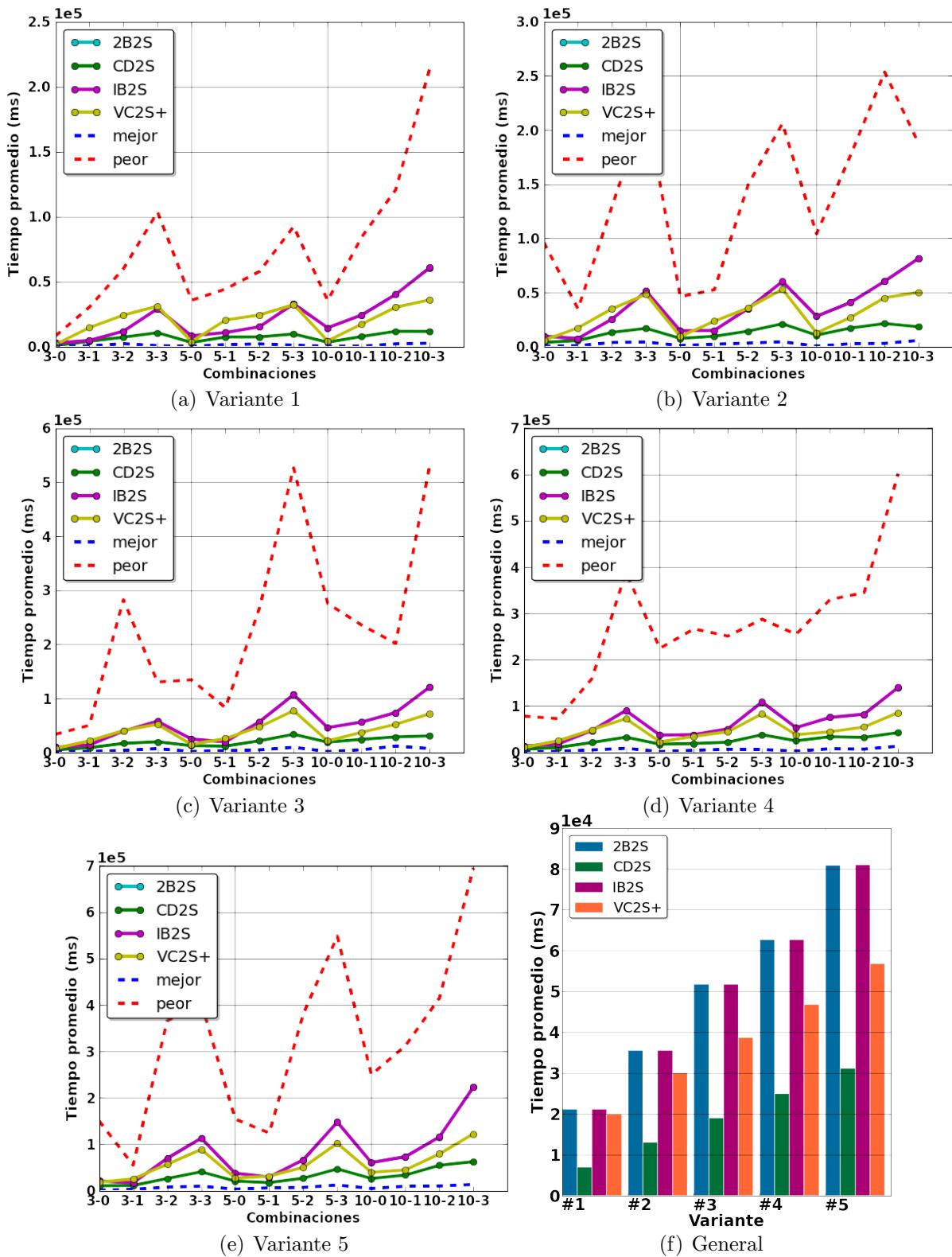


Figura 5.7: Tiempo de ejecución promedio por lote

El algoritmo VC2S+ en las combinaciones de 0 dimensiones no espaciales, debería tener un mejor desempeño que los demás, debido a que el algoritmo en su forma original acepta estos casos de prueba y poda el espacio de búsqueda si el vecino del vecino es dominado por otro elemento del *skyline* o por la pila de candidatos a *skyline*. Sin embargo, esto no es así, la línea correspondiente al algoritmo VC2S+, en las Figuras 5.7(a)-5.7(e), tiene sus puntos mínimos en las combinaciones 3 – 0, 5 – 0 y 10 – 0, al igual que el algoritmo CD2S, pero la línea de este último esta por debajo de VC2S+. Por lo tanto, según las pruebas realizadas, como el tiempo promedio de VC2S+ es mayor que el tiempo de CD2S en todos los casos, incluyendo las combinaciones de 0 dimensiones no espaciales, se **rechaza H2**.

5.3.2.2 Estudio de los patrones de desplazamiento y la actividad en el *skyline*

En cuanto a los patrones de cambio presentados en las simulaciones, en la Figura 5.8 de **casos de desplazamiento** ejecutados, donde el eje *y* muestra los posibles **casos** y el eje *x* la **frecuencia** en que ocurren en un lote, se puede ver que a medida que aumenta *DistMax*, disminuye la frecuencia de ejecución de los casos sencillos para la actualización del *skyline*: **caso 1** que indica que el algoritmo debe devolver el mismo conjunto de la simulación anterior, **caso 3** que se refiere a que sólo tiene que eliminar objetos del *skyline* y los **casos 2-4** que implica sólo agregar objetos faltantes al mismo. Así como también, con el aumento del parámetro *DistMax*, crece la frecuencia de los casos más complejos: **caso 5** que indica que algoritmo necesita agregar y eliminar objetos del *skyline* para su actualización y **otros casos**, donde CD2S realiza el mismo procedimiento que para el **caso 5**, pero para VC2S+ implica **reiniciar** las estructuras y calcular el *skyline*.

Para la variante 1 se tiene que un 12,2% de las simulaciones pertenecen al caso 1; 18,4% del caso 3; 22,4% de los casos 2-4; 36,7% del caso 5 y 6,1% para otros casos. Estos porcentajes cambian notablemente para la variante 5, disminuyendo la frecuencia de los casos del 1 al 4 a 0%, 10,2% y 12,2% respectivamente y aumentando a 49% y 22,4% los casos que implican una mayor cantidad de cambios en el *skyline* para su actualización.

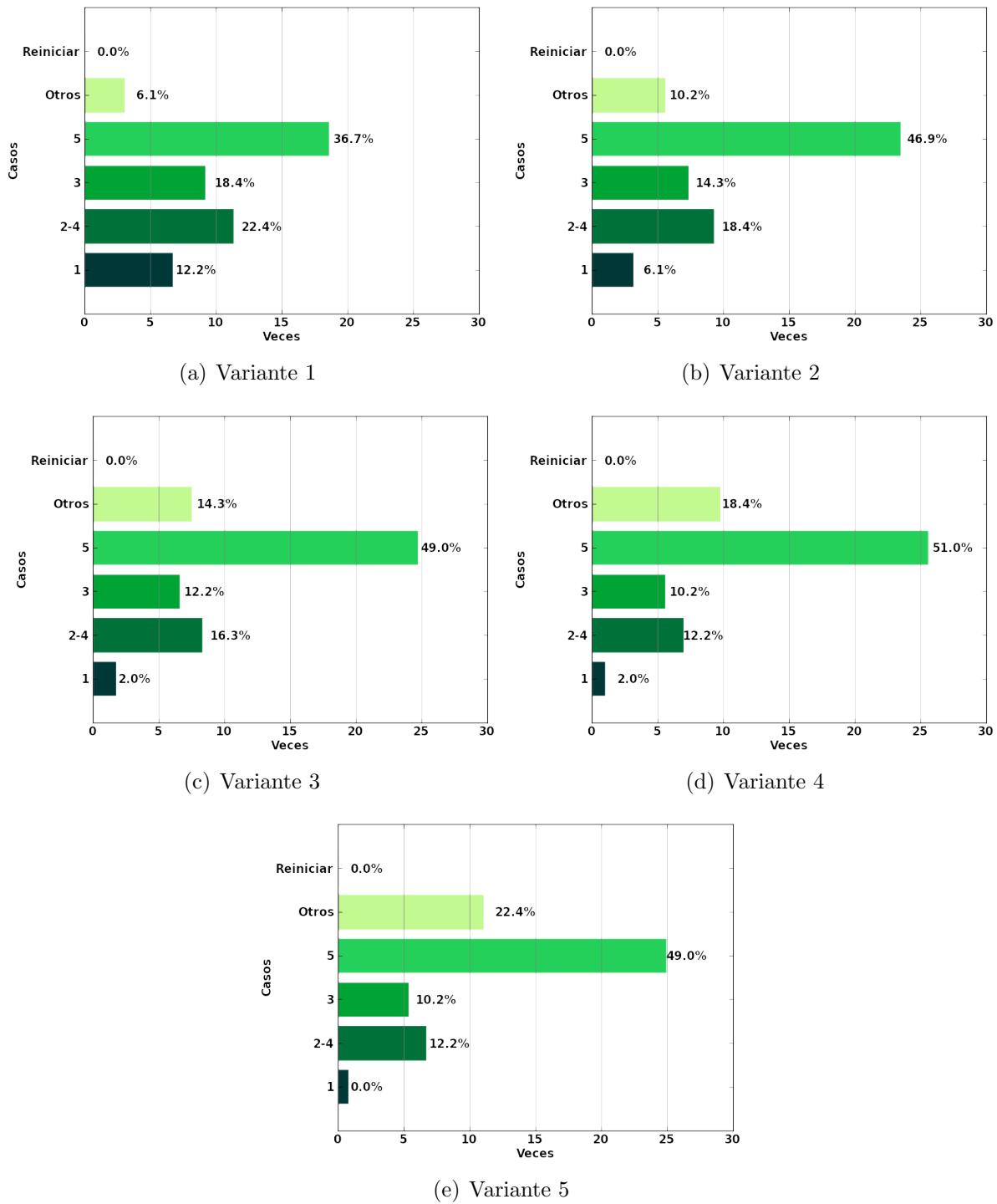


Figura 5.8: Casos de desplazamiento en promedio por lote

Adicionalmente, en la Figura 5.9 de cambios realizados, los cambios que deben hacer los algoritmos en el *skyline* para actualizarlo o construirlo aumentan mientras aumente el valor de *DistMax*. En general, es mayor la cantidad de elementos que se agregan al *skyline* que los que son eliminados.

Tomando en cuenta que a medida que aumenta el valor de *DistMax*, la frecuencia en los patrones del 1-4 disminuye y la cantidad de cambios u operaciones a ser realizados en el *skyline* para construir el conjunto final, aumenta, se **aprueba H3** para este experimento.

La cantidad de cambios que necesita realizar el algoritmo CD2S para actualizar el *skyline*, es menor en todas las variantes a los demás algoritmos, esto es porque resulta mas precisa la decisión que se toma para incluir, descartar o eliminar el objeto del *skyline* al etiquetar cada objeto del *skyline* según la porción a la que pertenece.

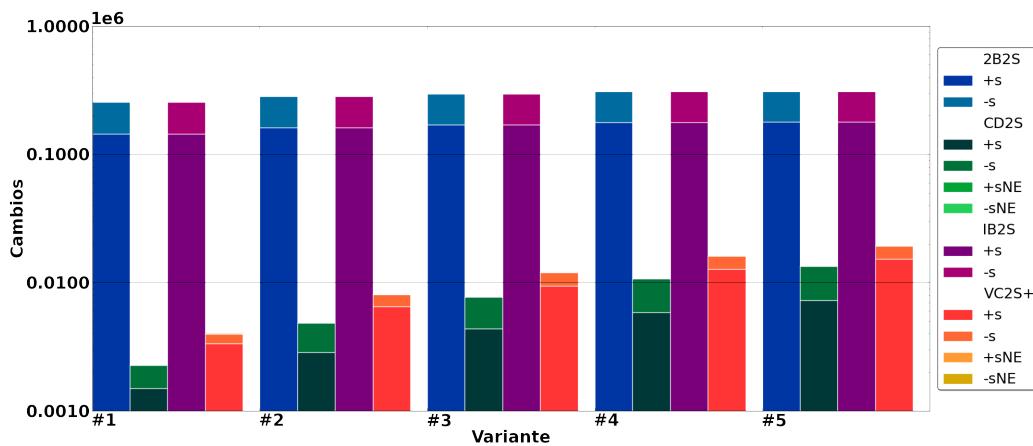


Figura 5.9: Cambios en el *skyline* en promedio por lote (Escala logarítmica)

El detalle de los resultados de estas simulaciones por combinación puede ser visto en el Apéndice E en las Tablas E.5, E.6, E.7, E.8 y E.9. Así como también, estudios más particulares pueden ser vistos en el Apéndice H.3.

5.4 Experimento IV

El objetivo de este experimento es estudiar el comportamiento de los algoritmos implementados únicamente en el *modo actualización de estatus*, respecto a la variación del porcentaje de cambio de disponibilidad de las localidades en cada simulación $\%Cambio$. Se intuye que una alta tasa del parámetro $\%Cambio$ implicará una mayor variación del *skyline* comparado con el obtenido en la simulación anterior por la cantidad de puestos que se verán afectados por el cambio, lo cuál se reflejará en el rendimiento de los algoritmos CD2S y IB2S que actualizan el *skyline* en este modo de ejecución.

Se tienen las siguientes hipótesis:

- H1** A mayor porcentaje de cambio de disponibilidad de localidades $\%Cambio$, menos diferencia existirá en el tiempo de ejecución promedio de los algoritmos CD2S y IB2S en comparación con el resto.
- H2** El porcentaje de cambio de disponibilidad $\%Cambio$ afecta directamente a la cantidad de cambios que deben realizar los algoritmos CD2S y IB2S para actualizar el *skyline*.

5.4.1 Metodología

Este experimento se realizó mediante los casos de prueba de la configuración 2 (Tabla 4.1, Capítulo IV) que son los mismos utilizados para el Experimento II y III, con cada variante se generaron 10 lotes para cada caso de prueba, en total 480 lotes, a través de las siguientes simulaciones:

- * **Variante 1:** Conf=2, Modo=*modo actualización de estatus* y $\%Cambio=10$.
- * **Variante 2:** Conf=2, Modo=*modo actualización de estatus* y $\%Cambio=30$.
- * **Variante 3:** Conf=2, Modo=*modo actualización de estatus* y $\%Cambio=aleatorio$.
- * **Variante 4:** Conf=2, Modo=*modo actualización de estatus* y $\%Cambio=70$.
- * **Variante 5:** Conf=2, Modo=*modo actualización de estatus* y $\%Cambio=90$.

Para la variante 3, se definió el parámetro `%Cambio=aleatorio`, porque probablemente lleve a resultados generales similares a si se fijara el parámetro en $\%Cambio = 50$, pero éste valor aleatorio generará una mayor entropía.

5.4.2 Resultados

La **media** de los tiempos de ejecución de los algoritmos en todas las variantes observable en la Tabla 5.4, es bastante menor que la **media** existente en los resultados del Experimento III Tabla 5.3.

Ésta es la consecuencia de la inexistencia de cambios en el *convex hull* por desplazamiento, como no existe desplazamiento y un aumento en el `%Cambio` no afecta la cardinalidad del *skyline*, la cardinalidad se mantiene constante para las cinco variantes.

Algoritmo	Variante 1			Variante 2			Variante 3			Variante 4			Variante 5		
	%Cambio=10			%Cambio=30			%Cambio=azar			%Cambio=70			%Cambio=90		
	μ_i	σ_i	$\frac{\mu_i}{\mu_4}$	μ_i	σ_i	$\frac{\mu_i}{\mu_4}$	μ_i	σ_i	$\frac{\mu_i}{\mu_4}$	μ_i	σ_i	$\frac{\mu_i}{\mu_4}$	μ_i	σ_i	$\frac{\mu_i}{\mu_4}$
(1) 2B2S	5.922	5.777	3,17	5.418	5.278	1,88	5.440	5.308	1,61	5.289	5.158	1,27	5.340	5.213	1,22
(2) IB2S	2.667	2.081	1,43	3.100	2.049	1,08	3.517	2.255	1,04	4.145	2.487	1,00	4.343	2.643	1,00
(3) VC2S+	4.251	2.017	2,28	4.500	1.819	1,57	4.750	1.785	1,41	5.066	1.743	1,22	5.168	1.767	1,19
(4) CD2S	1.868	1.338	1,00	2.875	2.008	1,00	3.372	2.376	1,00	4.162	2.828	1,00	4.347	3.032	1,00
#Cambios	62.720,00 obj			188.160,00 obj			312.371,42 obj			439.040,00 obj			564.480,00 obj		
#vecesEst	49 veces			49 veces			49 veces			49 veces			49 veces		

Tabla 5.4: Resultados por lotes de 50 simulaciones

5.4.2.1 Estudio del tiempo de los algoritmos

La razón $\frac{\mu_i}{\mu_4}$ entre la media de los tiempos de los algoritmos y el algoritmo CD2S de la Tabla 5.4, disminuye a medida que aumenta `%Cambio`, esto se debe a que la cantidad de objetos que pasan a estar disponibles u ocupados aumentan, por lo que son más los objetos que debe considerar el algoritmo CD2S en el *modo actualización de estatus* para actualizar el *skyline*.

Mas aún, cuando la cardinalidad de los objetos que cambiaron de estatus crece de 62.720 objetos en promedio en un lote en la variante 1 a 564.480 objetos en la variante 5.

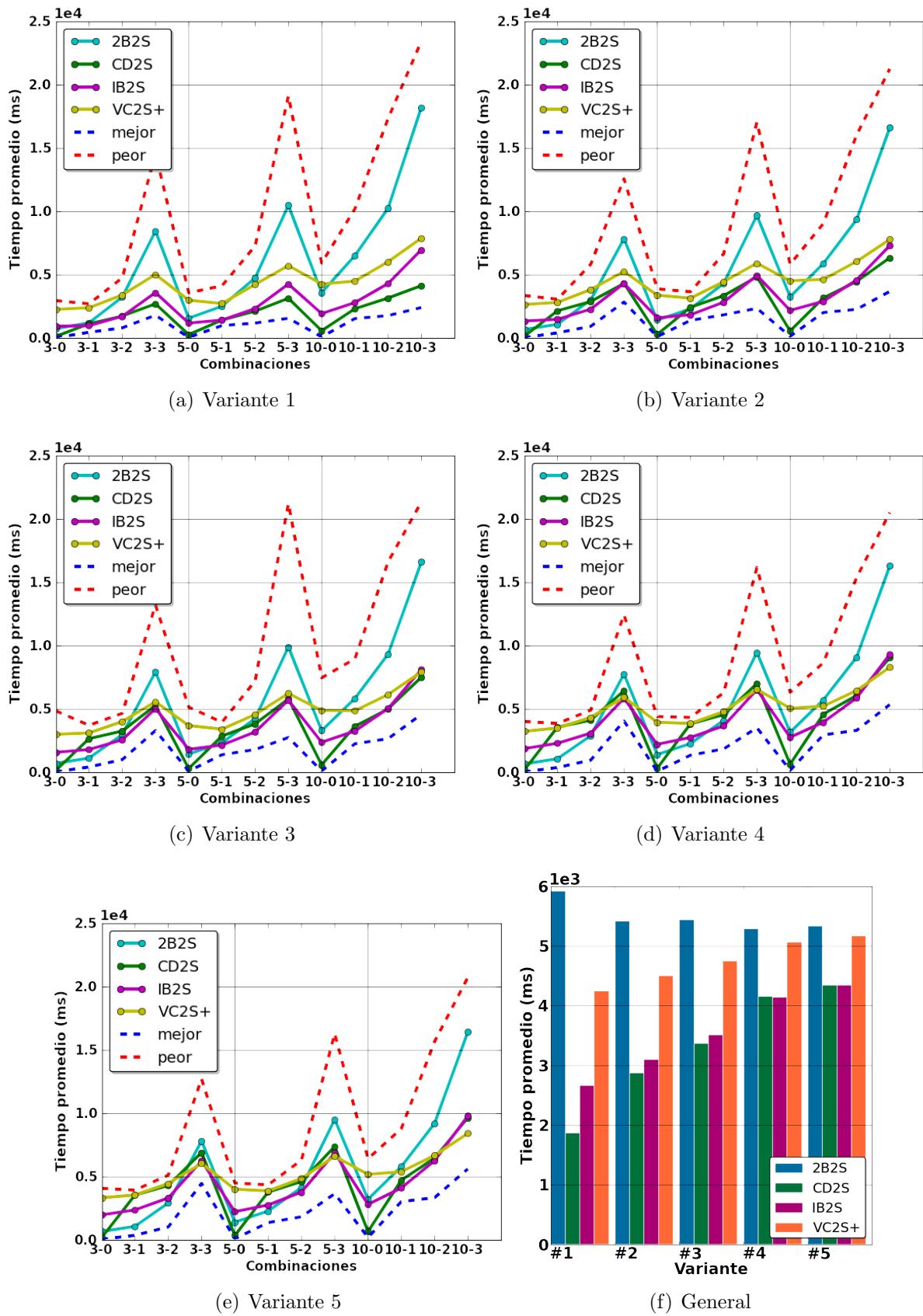


Figura 5.10: Tiempo promedio de ejecución por lote

Los algoritmos CD2S, IB2S, VC2S+ y 2B2S tienen como media de tiempo de ejecución en la variante 1: 1.868 ms, 2.667 ms, 4.251 ms y 5.922 ms y en la variante 5, cambia la media del tiempo de ejecución a: 4.347 ms, 4.343 ms, 5.168 ms y 5.340 ms. Lo que refleja una diferencia entre CD2S y los demás algoritmos que se va reduciendo de la variante 1 a la 5, siendo para la variante 1, la razón más alta, la correspondiente al algoritmo 2B2S con 3,17, la cual decrece a 1,22 para la variante 5. Este comportamiento, puede ser observado también en la Figura 5.10(f), la disminución de la diferencia del desempeño del algoritmo CD2S en relación a los demás algoritmos a medida que el %Cambio crece, es notable, lo cual **confirma H1**.

Como se observa en la Tabla 5.4, el tiempo promedio de ejecución del algoritmo IB2S para las variantes 4 y 5 es el mejor, lo cual probablemente sea el resultado de ejecutar un procedimiento más sencillo que el algoritmo CD2S para la actualización del *skyline* que se ve favorecido para un alto %Cambio.

Haciendo referencia a los tiempos promedio de ejecución en detalle por combinación en la Figura 5.10, se observa que existe una gran inestabilidad en los tiempos de los algoritmos. Las líneas se entrelazan siendo mejor: el algoritmo 2B2S en las combinaciones 3-1 de la variante 2 a la 5 y 3-2 y 5-1 de las variantes 4 y 5; el algoritmo IB2S para las combinaciones 3-2 y 5-1 de las variantes 2 y 3 y 5-2 y 10-1 de la variante 2 a la 5; el algoritmo VC2S+ para la combinación 10-3 de las variantes 4 y 5 y el algoritmo CD2S en el resto de las combinaciones, estando muy cerca el promedio de este último de la linea punteada azul correspondiente al mejor tiempo de los algoritmos en las combinaciones con 0 dimensiones no espaciales (3-0,5-0,10-0). Este comportamiento inusual, en comparación con los otros experimentos, se debe a que los algoritmos tienen un desempeño bastante similar para el modo de ejecución *modo actualización de estatus*.

5.4.2.2 Estudio del porcentaje de cambio y la actividad en el *skyline*

En relación al número de modificaciones que deben efectuar los algoritmos CD2S y IB2S para retornar el *skyline* correcto, en la gráfica de cambios en el *skyline* de la Figura 5.11, se puede apreciar que es considerablemente mayor a medida que se incrementa el

$\%Cambio$, siendo imperceptible para la variante 1 (menor a 1×10^4), lo cual **confirma** la hipótesis **H2** para este experimento. Este fenómeno se debe a que por cada cambio de estatus (disponible/ocupado) de los objetos, los algoritmos que actualizan el conjunto *skyline* deben revisar si el cambio implica una eliminación del objeto en el *skyline* o si por el contrario implica una inclusión del objeto y tal vez una o más eliminaciones de los objetos dominados por éste. En cambio, los algoritmos que calculan el *skyline* desde el inicio en cada simulación, no realizan estas modificaciones y por eso el número de cambios se mantiene constante.

En algunos casos resulta mejor computar el *skyline* completo que realizar una actualización del mismo, lo cual se refleja en las variantes 3, 4 y 5 de la Figura 5.11, donde es más eficiente el algoritmo **VC2S+** que el algoritmo **CD2S** ya que realiza una mejor escogencia de los elementos que probablemente formarán parte del *skyline*, teniendo como resultado una menor cantidad de cambios efectuados. Esta es la razón por la cual el algoritmo **VC2S+** se acerca más en el número de comparaciones de dominancia al algoritmo **CD2S**.

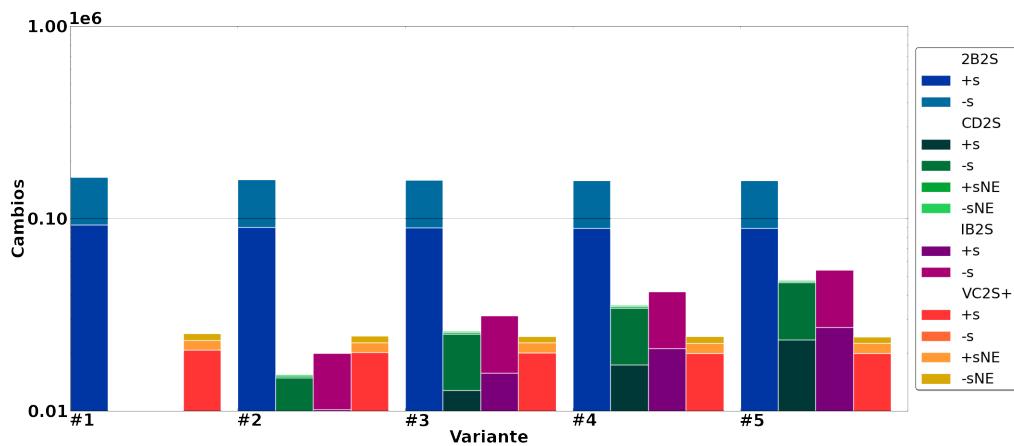


Figura 5.11: Cambios en el *skyline* en promedio por lote (Escala logarítmica)

El detalle de los resultados de estas simulaciones por combinación ser visto en el Apéndice E en las Tablas E.10, E.11, E.12, E.13 y E.14. Así como también, estudios más particulares pueden ser vistos en el Apéndice H.4.

5.5 Experimento V

El objetivo de este experimento es estudiar el comportamiento de los algoritmos implementados respecto a la variación del tamaño máximo del *convex hull* inicial $CHMax$, utilizando un conjunto de datos de mayor cardinalidad que el Experimento II.

Se tienen las siguientes hipótesis:

- H1** El tiempo de ejecución del algoritmo CD2S es menor que el tiempo del resto de los algoritmos
- H2** A mayor región inicial para situar los puntos de referencia $CHMax$, mayor cardinalidad del *skyline* y tiempo de ejecución.

5.5.1 Metodología

Este experimento se realizó mediante los casos de prueba de las configuraciones 4 ($CHMax = 10\%$), 5 ($CHMax = 25\%$) y 6 ($CHMax = 50\%$) (Tabla 4.1, Capítulo IV), con cada variante se generaron 10 lotes para cada caso de prueba, en total 240 lotes, a través de las siguientes simulaciones:

- * **Variante 1:** Conf=4, Modo=combinado, %Cambio=aleatorio y DistMax=10.
- * **Variante 2:** Conf=5, Modo=combinado, %Cambio=aleatorio y DistMax=10.
- * **Variante 3:** Conf=6, Modo=combinado, %Cambio=aleatorio y DistMax=10.

5.5.2 Resultados

En la Tabla 5.5, la cantidad de objetos que cambiaron de estatus, #Cambios, es similar para las 3 variantes, al igual que la distancia recorrida en promedio por el punto de referencia en un lote, Desplazamiento, y los modos de simulación #vecesEst y #vecesMov, esto es porque las variantes del experimento, únicamente se diferencian en el parámetro que limita el tamaño del *convex hull*.

Algoritmo	Variante 1 CHMax=10 %			Variante 2 CHMax=25 %			Variante 3 CHMax=50 %		
	μ_i	σ_i	$\frac{\mu_i}{\mu_4}$	μ_i	σ_i	$\frac{\mu_i}{\mu_4}$	μ_i	σ_i	$\frac{\mu_i}{\mu_4}$
	(1) 2B2S	123.089	124.934	1,78	1.544.995	1.568.596	2,46	9.216.985	10.335.941
(2) IB2S	90.022	78.132	1,30	1.035.860	1.131.562	1,65	6.408.713	7.610.420	1,59
(3) VC2S+	112.067	76.400	1,62	1.211.269	1.353.082	1,93	6.749.080	8.278.597	1,67
(4) CD2S	69.193	57.001	1,00	627.646	710.732	1,00	4.036.858	5.447.505	1,00
Desplazamiento	135,00 m			130,58 m			137,50 m		
#Cambios	631.769,50 obj			643.698,25 obj			616.551,25 obj		
#vecesEst	24,41 veces			24,92 veces			24,00 veces		
#vecesMov	24,58 veces			24,08 veces			25,00 veces		

Tabla 5.5: Resultados por lotes de 50 simulaciones

5.5.2.1 Estudio del tiempo de los algoritmos

Se observa la Tabla 5.5 un mejor desempeño, en tiempo de ejecución, para el algoritmo CD2S cuando $CHMax = 25 \%$, donde la diferencia entre las medias de los algoritmos, $\frac{\mu_i}{\mu_4}$, es mayor, siendo la más alta para 2B2S con 2,46 y la menor para IB2S con 1,65.

En general, se evidencia que la media del tiempo de ejecución μ en todas las variantes, también observada en la Figura 5.12(a), es menor para CD2S en comparación con el resto de los algoritmos, al igual que la desviación estándar σ_i , por lo que se **confirma H1**.

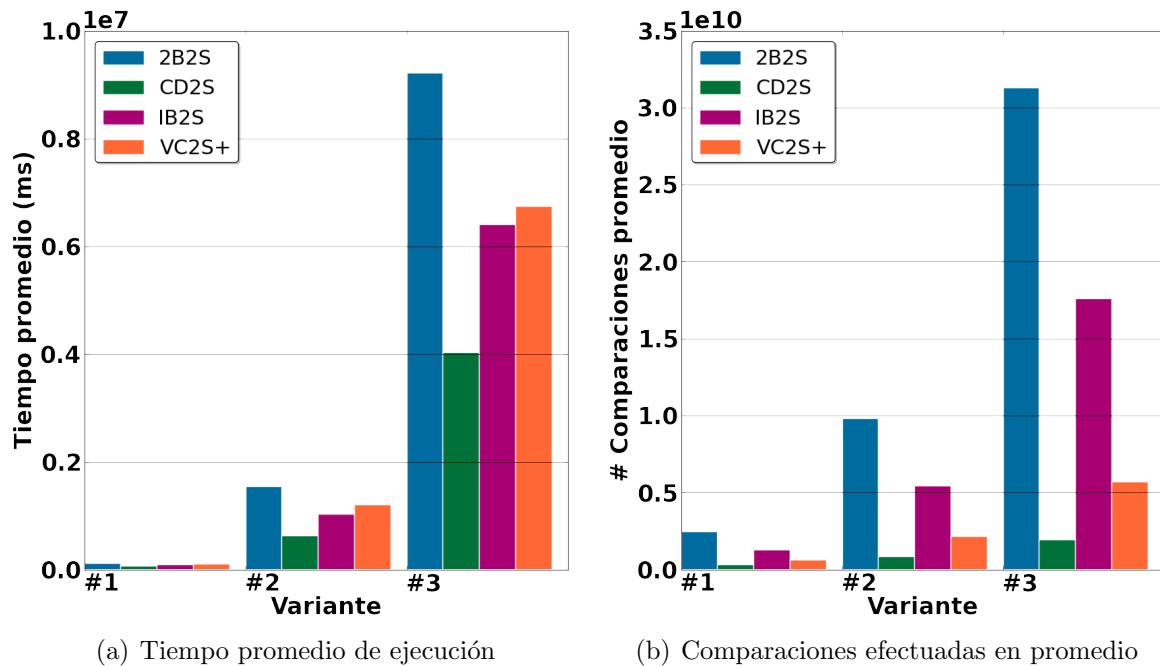


Figura 5.12: Resultados generales

El algoritmo CD2S tiene un mejor desempeño que los demás algoritmos, porque realiza un número menor de comparaciones de dominancia para devolver el *skyline* correcto, lo cual puede ser visto en la Figura 5.12(b).

5.5.2.2 Estudio del área del *convex hull*

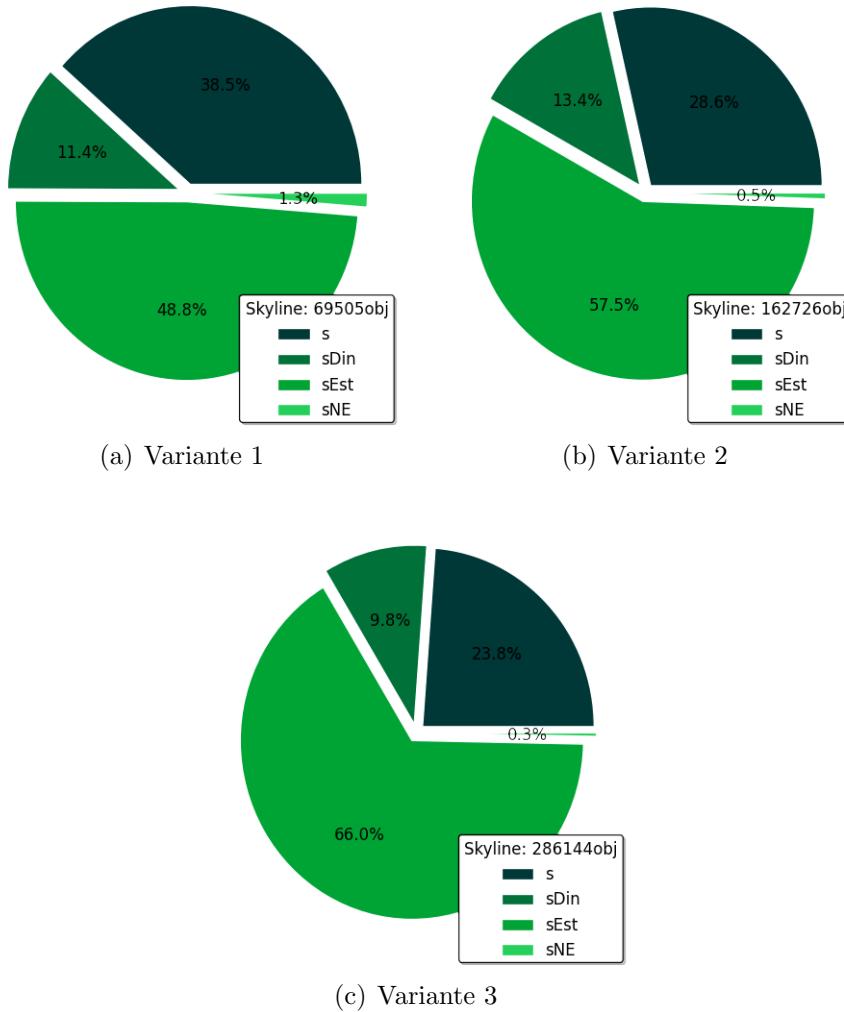
El área del *convex hull* inicial, afecta directamente al tiempo de ejecución empleado por los algoritmos para obtener el *skyline*. En la Figura 5.12(a), se constata que el tiempo promedio aumenta a medida que crece *CHMax*.

Naturalmente, el incremento en el tiempo de ejecución, se debe a que la cardinalidad del *skyline* en la Figura 5.13 crece al aumentar el parámetro *CHMax*. Donde, la cardinalidad del *skyline* promedio cambia de 69.505 objetos para la variante 1 a 286.144 objetos para la variante 3.

Debido a que el área del *convex hull* inicial, afecta la cardinalidad del *skyline* y esta a su vez el tiempo de ejecución de los algoritmos, a un mayor valor de *CHMax*, mayor será la cantidad de objetos en el *skyline* y el tiempo empleado por los algoritmos para conseguirlos y retornarlos, lo cual **confirma** la hipótesis **H2**.

El área del *convex hull* inicial, afecta también, en la Figura 5.13, la proporción correspondiente al *skyline* estático de 48,8 % a 66 %. Esto hace que el número de comparaciones de dominancia en la Figura 5.12(b), para el algoritmo CD2S sea menor que para el resto de los algoritmos, porque el algoritmo se aprovecha de la clasificación del *skyline* para reducir las comparaciones a efectuar.

El detalle de los resultados de estas simulaciones por combinación puede ser visto en el Apéndice E en las Tablas E.17, E.18 y E.19. Así como también, estudios más particulares pueden ser vistos en el Apéndice H.5.

Figura 5.13: Composición promedio del *skyline* por lote

5.6 Experimento VI

El objetivo de este experimento es estudiar el comportamiento de los algoritmos implementados utilizando un mayor número de puntos de referencia que los experimentos anteriores y un conjunto de datos de alta cardinalidad. Al mismo tiempo, se desea analizar la influencia de las dimensiones no espaciales en este comportamiento, por lo que se separó el estudio para 0 y 1 dimensiones no espaciales. El algoritmo VC2S+ con los casos de prueba de 0 dimensiones no espaciales, debería verse favorecido por la forma en que poda los objetos a visitar para obtener el *skyline*. Se excluyó el algoritmo 2B2S de este estudio, ya que se ha observado en los experimentos anteriores que requiere de un mayor tiempo para conseguir el *skyline*.

Se tienen las siguientes hipótesis:

- H1** El tiempo de ejecución de los algoritmos aumenta con el número de puntos de referencia.
- H2** A mayor número de puntos de referencia, mayor número de comparaciones y cardinalidad del *skyline*.
- H3** El algoritmo VC2S+ realiza un menor número de comparaciones para los casos de 0 dimensiones no espaciales porque debe verificar la dominancia de menos objetos.
- H4** Los candidatos al *skyline* aumentan para los algoritmos VC2S+ y CD2S en presencia de dimensiones no espaciales.

5.6.1 Metodología

Este experimento se realizó mediante los casos de prueba de la configuración 7 ($CHMax = 10\%$) (Tabla 4.1, Capítulo IV), los cuales se separaron en dos conjuntos: los casos de prueba de las combinaciones 12-0,14-0,16-0 y 18-0 (configuración 7a) y los casos de las combinaciones con una dimensión espacial, 12-1,14-1,16-1 y 18-1 (configuración 7b).

Con cada variante se generaron 10 lotes para cada caso de prueba, en total 160 lotes, a través de las siguientes simulaciones:

- * **Variante 1:** Conf=7a, Modo=combinado, %Cambio=aleatorio y DistMax=50 .
- * **Variante 2:** Conf=7b, Modo=combinado, %Cambio=aleatorio y DistMax=50.

5.6.2 Resultados

En general, el algoritmo CD2S tiene el menor tiempo de ejecución. Se observa en la Tabla 5.6 que la **media** para la variante 1 es 2,20 y 1,53 veces menor que las medias de los algoritmos VC2S+ y IB2S, al igual que se observa que la **desviación estándar** es menor. De la misma manera, la media de CD2S para la variante 2 es 1,75 y 1,49, veces menor que las medias de los algoritmos VC2S+ y IB2S. Se observa una mayor media en esta variante para todos los algoritmos el cual es el resultado de tener una dimensión más en el *skyline*.

Algoritmo	Variante 1			Variante 2		
	μ_i	σ_i	$\frac{\mu_i}{\mu_4}$	μ_i	σ_i	$\frac{\mu_i}{\mu_4}$
(1) 2B2S	-	-	-	-	-	-
(2) IB2S	1.814.639	1.190.522	1,53	2.627.835	2.175.205	1,49
(3) VC2S+	2.598.829	1.766.741	2,20	3.094.533	2.655.859	1,75
(4) CD2S	1.183.134	796.704	1,00	1.769.436	1.396.894	1,00
Desplazamiento	601 m			623 m		
#Cambios	1.262.073,75 obj			1.204.053,00 obj		
#vecesEst	24,5 veces			24,5 veces		
#vecesMov	24,5 veces			24,5 veces		

Tabla 5.6: Resultados por lotes de 50 simulaciones

5.6.2.1 Estudio del tiempo de los algoritmos

En lo referente al tiempo de ejecución por combinación, en las Figuras 5.14(a) y 5.14(b), se aprecia en ambas variantes, un aumento en los tiempos de ejecución de los algoritmos a medida que se agregan más dimensiones.

Estos tiempos cambian en la variante 1, en la combinación 12-0 de $10,8 \times 10^5 ms$, $16,7 \times 10^5 ms$ y $7,6 \times 10^5 ms$ a $26,2 \times 10^5 ms$, $36,7 \times 10^5 ms$ y $16,4 \times 10^5 ms$ en la combinación 18-0, para los algoritmos IB2S, VC2S+ y CD2S respectivamente. Similarmente, en la variante 2,

estos valores cambian en la combinación 12-1 de $16,2 \times 10^5 ms$, $20,1 \times 10^5 ms$ y $11,5 \times 10^5 ms$ a $32,4 \times 10^5 ms$, $37,6 \times 10^5 ms$ y $21,4 \times 10^5 ms$ en la combinación 18-1.

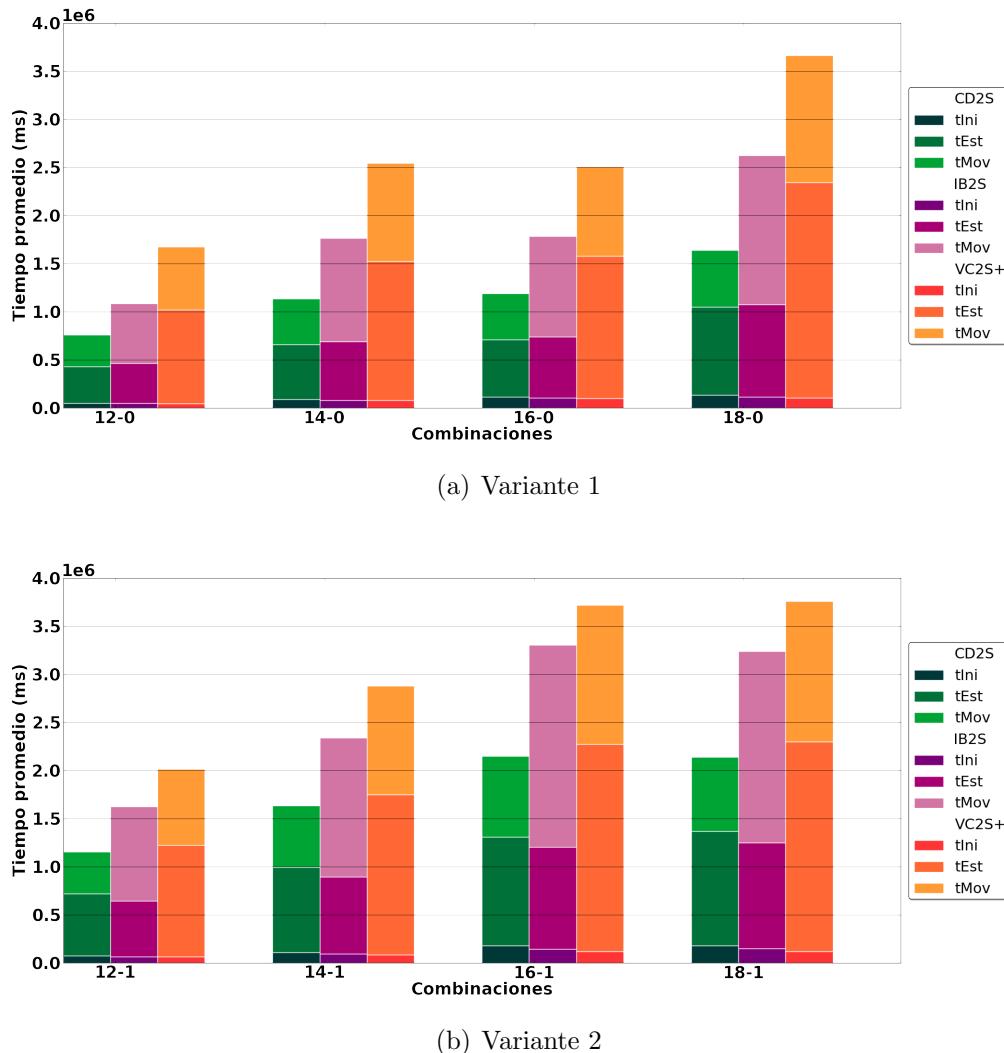


Figura 5.14: Tiempo promedio por lote y modo de ejecución

Por lo tanto, se observa claramente un incremento en los tiempos de ejecución de ambas variantes, a medida que aumenta el número de puntos de referencia, es por esto que se **aprueba H1** para el estudio realizado.

Esta consecuencia del aumento de puntos de referencia se percibe también, en la Figura 5.14, tanto en la fracción del tiempo empleado en el *modo actualización de estatus tEst* como en el *modo actualización de desplazamiento tMov*, lo que quiere decir que ambos modos de ejecución se ven afectados por un mayor número de puntos de referencia.

Cabe destacar, que el valor de t_{Mov} para el algoritmo VC2S+, en todas las combinaciones y variantes, es menor o similar que el valor del IB2S. A pesar de esto, VC2S+ requiere más del doble del tiempo t_{Est} que IB2S. Ambos, aspectos estudiados en los Experimentos III y IV.

5.6.2.2 Estudio de las comparaciones y cardinalidad del *skyline*

El número de comparaciones efectuadas, presentes en la Figuras 5.15(a) y 5.15(b), se ve afectado también por las dimensiones no espaciales o puntos de referencia en el *skyline*. Las comparaciones cambian en la variante 1, en la combinación 12-0 de $7,5 \times 10^9 ms$, $1,3 \times 10^9 ms$ y $0,8 \times 10^9 ms$ a $19,1 \times 10^9 ms$, $2 \times 10^9 ms$ y $1,2 \times 10^9 ms$ en la combinación 18-0, para los algoritmos IB2S, VC2S+ y CD2S respectivamente. Similarmente, estos valores cambian en la combinación 12-1 de $10 \times 10^9 ms$, $2,8 \times 10^9 ms$ y $1,3 \times 10^9 ms$ a $22,3 \times 10^9 ms$, $3,7 \times 10^9 ms$ y $1,8 \times 10^9 ms$ en la combinación 18-1. Evidentemente, se realizan más comparaciones en la variante 2 por tener además una dimensión no espacial.

Existe una ligera disminución del número de comparaciones realizadas por el algoritmo VC2S+ en las combinaciones 16-0 y 18-1, el cual debe ser el comportamiento de algún caso particular que se haya generado.

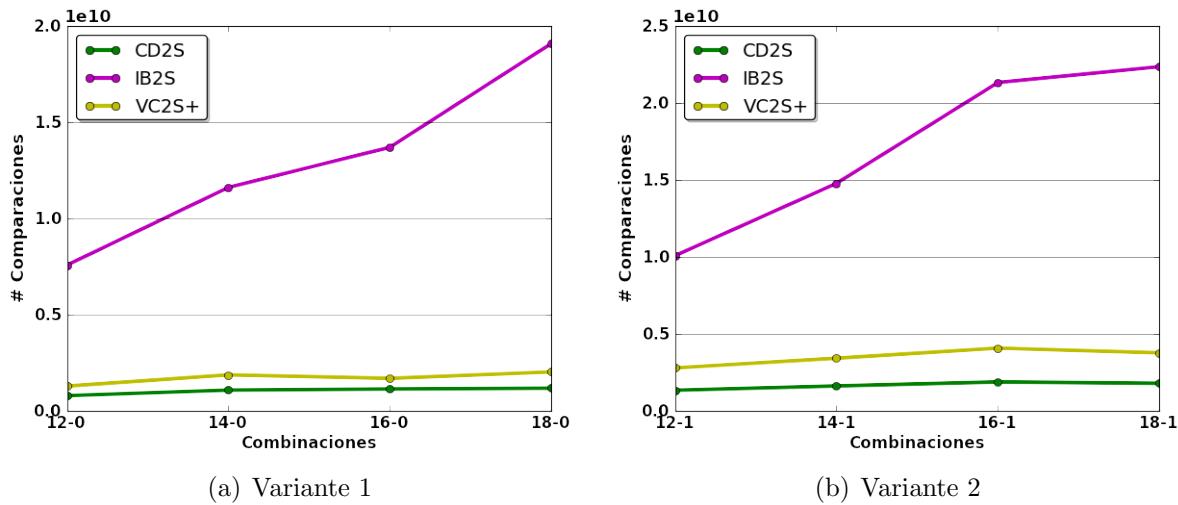


Figura 5.15: Comparaciones efectuadas en promedio

En cuanto a la cardinalidad del *skyline*, se evidencia en las Figuras 5.16(a) y 5.16(b), un aumento en la cardinalidad a medida que aumentan estas dimensiones espaciales. En la variante 1 el *skyline* cambia de 177.879 objetos para la combinación 12-0 a 238.735 objetos por para la combinación 18-0 y en la variante 2 de 190.920 objetos para la combinación 12-1 a 244.130 objetos para la combinación 18-1.

Debido a que el número de puntos de referencia, afecta tanto el número de comparaciones que deben realizar los algoritmos como en la cardinalidad del *skyline* encontrado, haciendo que sean mayores estos valores al existir una mayor cantidad de puntos de referencias, se aprueba **H2**.

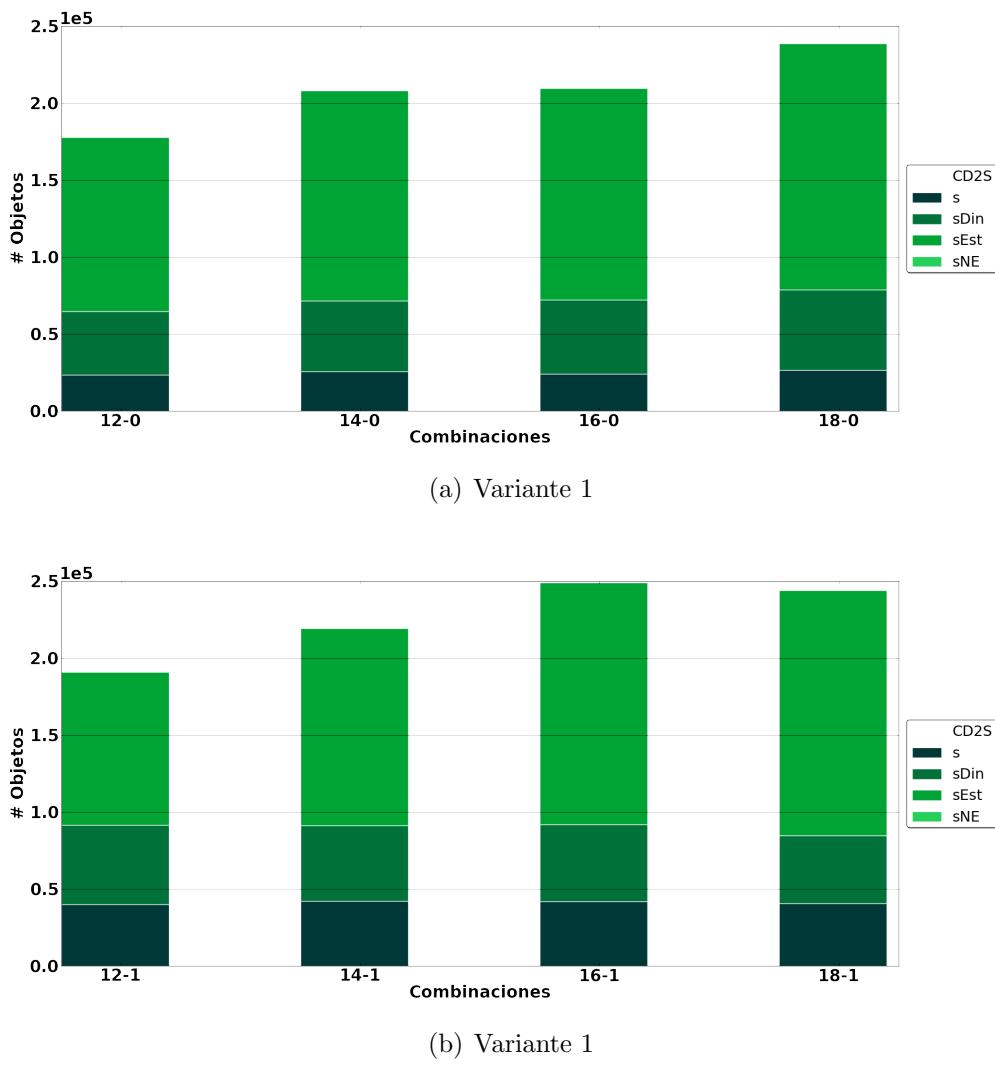


Figura 5.16: Composición promedio del *skyline* por lote

Por otro lado, se observa en la Figura 5.15, que el menor número de comparaciones en ambas variantes y en todas las combinaciones corresponde al algoritmo CD2S, en lugar del algoritmo VC2S+.

Aunque las líneas de estos algoritmos están cerca, se esperaba que VC2S+ realizará la menor cantidad de comparaciones para la variante 1. El algoritmo VC2S+ para los casos de cero dimensiones no espaciales, necesita visitar a una menor cantidad de objetos para obtener el *skyline*, ya que puede aplicar su heurística y detener la búsqueda a tiempo, como se observa en la Figura 5.17(a).

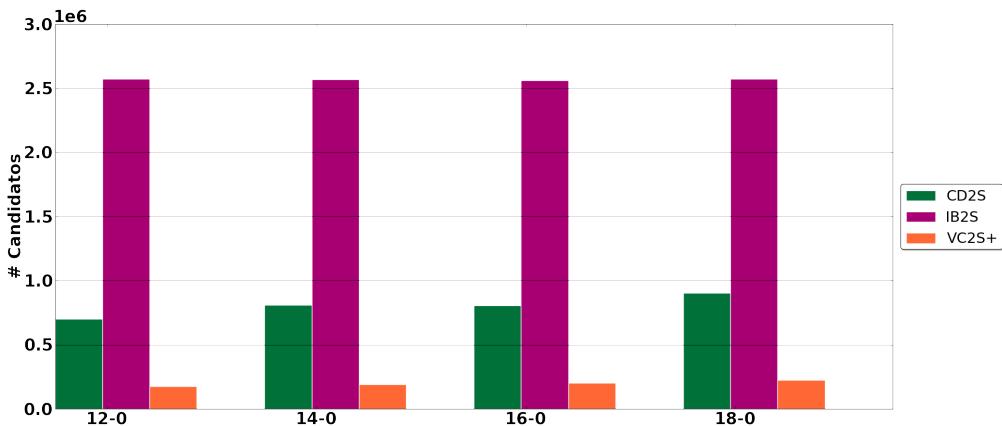
Sin embargo, el número de comparaciones del algoritmo VC2S+ no es el menor, por lo que se **rechaza H3**. Por ser alta la proporción del *skyline* estático en la Figura 5.16, p. 120, el algoritmo CD2S realiza el menor número de comparaciones, ya que menos objetos deben ser evaluados para actualizar el *skyline*.

5.6.2.3 Estudio de los candidatos al *skyline*

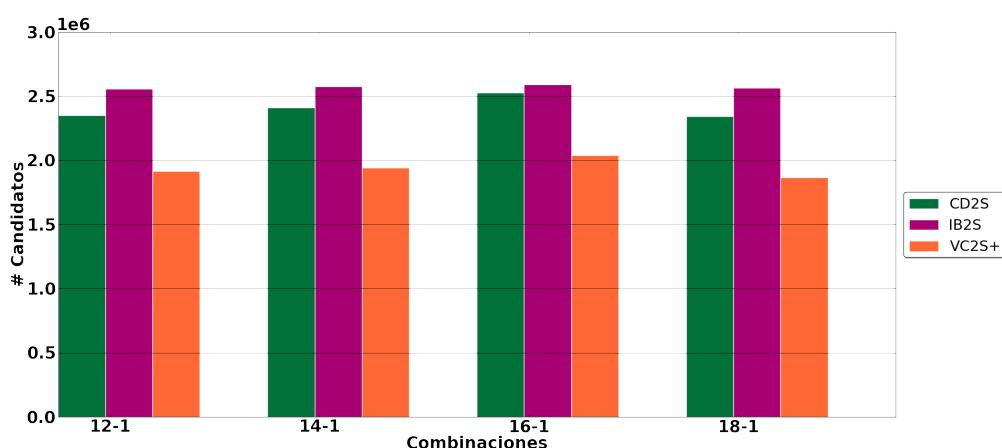
En las Figuras 5.17(a) y 5.17(b), se pueden percibir notablemente una mayor cantidad de candidatos para los algoritmos VC2S+ y CD2S para la variante 2, en comparación con la variante 1, en tal sentido, se **aprueba H4**. En la variante 2, el número de candidatos al *skyline* que tienen estos dos algoritmos, se acerca a la cantidad que revisa el algoritmo IB2S.

Esto se debe a que como las dimensiones no espaciales, no están relacionadas con las espaciales, los objetos del *skyline* no espacial pueden estar en cualquier lugar del plano. Como VC2S+ y CD2S toman los objetos del *skyline* no espacial para construir la región de búsqueda (Figura C.1, p. 145), la región de búsqueda debe ser mayor a las que hayan sido generadas en la variante 1. Esto se debe a que, lo más alejado que podría estar un elemento de los puntos de referencia para formar parte del *skyline*, es hasta la misma distancia que están los objetos del *skyline* no espacial.

La cantidad de candidatos para el algoritmo IB2S permanece constante en ambas variantes porque este algoritmo revisa todos los puestos disponibles. El algoritmo VC2S+ visita la menor cantidad de candidatos en ambas variantes, sin embargo para el algoritmo CD2S se cuentan como candidatos los objetos dentro del *convex hull*, pero una gran parte de estos pertenecen al *skyline* estático.



(a) Variante 1



(b) Variante 2

Figura 5.17: Candidatos al *skyline* en promedio por lote

El detalle de los resultados de estas simulaciones por combinación puede ser visto en el Apéndice E en la Tabla E.20. Así como también, estudios más particulares pueden ser vistos en el Apéndice H.6.

CAPÍTULO VI

CONCLUSIONES Y RECOMENDACIONES

Este trabajo de grado se basó en el uso de las consultas *skyline* [3] para la especificación de preferencias sobre objetos espaciales. En tal sentido, se planteó el problema más específico de *skyline* espacial sobre datos dinámicos *dynamic spatial skyline* (DSS, por sus siglas en inglés), el cual consiste en la inclusión de estas preferencias sobre datos con atributos espaciales, no espaciales y, un requisito obligatorio sobre un atributo altamente cambiante. Las preferencias espaciales puedan ser establecidas sobre un conjunto de puntos de referencia, con la finalidad de conseguir los objetos más cercanos, al mismo tiempo que, uno de estos puntos se encuentra en movimiento; las preferencias no espaciales son definidas, al igual que las consultas *skyline* tradicionales, sobre atributos numéricos, cualitativos o categorías (previamente traducidas a una forma numérica); y el requisito obligatorio, es una condición que deben cumplir los objetos, por encima de las preferencias, para ser habilitados como posibles candidatos al conjunto *skyline*.

Tal problema surge por la necesidad de proporcionarle a los usuarios de sistemas de recomendación, un mecanismo capaz de filtrar los mejores objetos dentro de un conjunto de datos no estáticos, basado en la ubicación del usuario y un conjunto de posibles puntos interesantes y preferencias cualitativas.

Se estudiaron diferentes modelos de consultas *skyline*, en la búsqueda de alguno que pudiese ser aplicado al problema DSS. Los enfoques que más se aproximan a este problema [7, 20, 47, 58, 59], reciben un conjunto de puntos de referencia, pero no emplean preferencias no espaciales y menos preferencias sobre un atributo no espacial cambiante. Posteriormente, en [51], los autores incluyen en sus algoritmos, atributos no espaciales, pero

son sólo estáticos. Por otro lado, existen consultas que incluyen preferencias no espaciales sobre objetos espaciales [8, 45, 53, 80], pero las preferencias espaciales son reducidas a un sólo punto de referencia y tampoco consideran preferencias sobre un atributo no espacial dinámico. En consecuencia, debido a la inexistencia de un tipo de consultas *skyline* que cumpla con los parámetros descritos en el problema planteado, son adaptadas las consultas *skyline* espaciales *spatial skyline queries*, (SSQ, por sus siglas en inglés) [7] en este trabajo de grado, para que contemplen todas las características del problema DSS. A esta adaptación, se le denominó consultas *skyline* espaciales sobre datos dinámicos *dynamic spatial skyline queries*, (DSSQ, por sus siglas en inglés).

Para las consultas DSSQ, se propusieron cuatro algoritmos de evaluación, denominados 2B2S, IB2S, VC2S+ y CD2S. Estos algoritmos, evalúan de forma continua estas consultas a partir de las transformaciones del ambiente, generadas por el desplazamiento de un punto de referencia o por los nuevos valores del atributo cambiante. Los algoritmos 2B2S e IB2S fueron diseñados como técnicas ingenuas, diferenciándose entre sí, en que el algoritmo 2B2S no actualiza el conjunto *skyline* y, el algoritmo IB2S lo actualiza, según las transformaciones generadas por los nuevos valores de los objetos. El algoritmo VC2S+ es una adaptación de los algoritmos VS² y VCS² [1], para la inclusión de preferencias no espaciales (cuyos cambios son sugeridos por los autores) y el requisito obligatorio sobre el atributo cambiante. Este algoritmo, actualiza el *skyline* (en la mayoría de los casos) para las transformaciones ocurridas por el desplazamiento de un punto de referencia. Por último, el algoritmo CD2S evalúa de forma continua estas consultas a la vez que actualiza el *skyline* para ambas transformaciones, por lo que constituye una solución mejorada.

Entre estos algoritmos implementados, el algoritmo 2B2S es el más básico, porque con cada transformación del ambiente debe reiniciar el cálculo del *skyline* para obtener el conjunto de objetos correcto. Este algoritmo, compara todos los objetos del conjunto de datos entre sí para conseguir el *skyline*. Aunque 2B2S es ineficiente, fue implementado como base para comprobar que los demás algoritmos reducen el número de comparaciones de dominancia u objetos comparados; y por otro lado, para validar que el conjunto *skyline* retornado por los demás algoritmos, resultante de alguna actualización, sea el correcto.

Como mejora al 2B2S, se propuso el algoritmo IB2S. Este algoritmo, permite la actualización del *skyline* al ocurrir alteraciones en los datos. Tal actualización, reutiliza el conjunto *skyline* del ambiente antes de ser transformado (consulta anterior), porque elimina de ese conjunto los objetos que pasaron a estar prohibidos y, si ninguno resultó prohibido, sólo verifica la dominancia de los nuevos objetos habilitados y elimina los objetos dominados; de lo contrario, verifica la dominancia de todos los objetos, pero contra ese conjunto *skyline*. En caso de desplazamiento del punto de referencia móvil, este algoritmo es idéntico al 2B2S.

Una mejoría a las soluciones ingenuas con respecto al desplazamiento del punto de referencia móvil, es proporcionada por el algoritmo VC2S+. Este algoritmo, emplea un diagrama de Voronoi para almacenar los objetos, forma un *convex hull* con los puntos de referencia y actualiza el *skyline* en cinco de los seis posibles patrones de cambio del *convex hull*. Además, evalúa el *skyline* de forma progresiva, haciendo que los objetos pertenecientes al *convex hull* sean agregados directamente (sin verificación) al conjunto *skyline*. El *skyline* no espacial formado por las preferencias no espaciales, es procesado aparte, en la consulta inicial y al ocurrir transformaciones por los nuevos valores de los objetos. En presencia de estas transformaciones, el algoritmo debe procesar el *skyline* desde el inicio porque su representación del ambiente no permite cambios de estatus en los objetos.

Como una solución al problema de evaluación progresiva de estas consultas, fue propuesto el algoritmo CD2S. Este algoritmo dispone de un *convex hull* principal, formado por todos los puntos de referencia y otro contenido por éste, que sólo incluye los puntos de referencia estáticos, *convex hull* estático. Este algoritmo, evalúa el *skyline* progresivamente y lo actualiza, a través de la separación del *skyline* en cuatro grupos: el *skyline* no espacial (denotado **sNE**) procedente de las preferencias no espaciales, el *skyline* estático (denotado **sEst**) formado por los objetos dentro del *convex hull* estático, el *skyline* dinámico (denotado **sDin**) compuesto por los objetos dentro del *convex hull* principal y fuera del estático y, el *skyline* fuera del *convex hull* principal (denotado **s**).

El algoritmo CD2S al igual que el VC2S+, incluye directamente en el *skyline*, los objetos dentro del *convex hull*. A diferencia del algoritmo VC2S+, el CD2S, separa los objetos del *convex hull* en los conjuntos **sEst** y **sDin**, ya que al ocurrir una transformación por desplazamiento, el *convex hull* estático sigue teniendo la misma forma. Como los conjuntos **sNE** y **sEst**, permanecen sin cambios al originarse esta transformación, no tienen que ser revisados por el algoritmo para confirmar pertenencia de estos objetos al *skyline*. Más aún, el algoritmo CD2S realiza optimizaciones en las verificaciones según el caso dado y el conjunto *skyline* al que pertenece el objeto de la ventana, para reducir el número de comparaciones necesarias en la verificación de dominancia de un candidato.

Tanto el algoritmo VC2S+ como CD2S, en presencia de dimensiones no espaciales, deben explorar el espacio de búsqueda completo para construir el *skyline* no espacial (sólo en la consulta inicial para CD2S). Para el resto de los objetos, ambos algoritmos utilizan mecanismos que permiten restringir la búsqueda de los objetos interesantes.

Los algoritmos propuestos, fueron probados con datos sintéticos mediante simulaciones. Para estas pruebas se realizaron seis experimentos variando diferentes parámetros del ambiente y de las simulaciones. En el Experimento I se estudió el comportamiento general de los algoritmos; en el Experimento II se analizó cómo influye el área del *convex hull* inicial en la evaluación del *skyline*; en los Experimentos III y IV se observó por separado, el comportamiento de los algoritmos para transformaciones del ambiente por desplazamiento del punto de referencia, únicamente, y por modificaciones en los valores del atributo cambiante, respectivamente; en el Experimento V se hizo un estudio similar al del Experimento II, pero con una cardinalidad mayor de los objetos; y en el Experimento IV se probaron los algoritmos con una cardinalidad alta de objetos y un mayor número de puntos de referencia; en estos experimentos se observó la diferencia entre especificar o no, dimensiones no espaciales. Para este último experimento se excluyó el algoritmo 2B2S por ser ineficiente en los experimentos anteriores.

Entre los resultados generales se observó que la cardinalidad del *skyline* aumenta a medida que se incrementan los puntos de referencia y dimensiones no espaciales. Así como también,

el área del *convex hull* y la distancia máxima a desplazarse el punto de referencia, influyen en los tiempos de ejecución y la cardinalidad del *skyline*. La cardinalidad de candidatos en el *skyline* para los algoritmos VC2S+ y CD2S aumenta en presencia de dimensiones no espaciales, así como el tiempo de ejecución. También se encontró que mientras más alta sea la tasa de cambio de los datos, mayor será la cantidad de modificaciones que deben realizar los algoritmos IB2S y CD2S para la actualización del *skyline* y menos diferencia existirá con los algoritmos 2B2S y VC2S+ que no lo actualizan en ese caso.

Con respecto a los algoritmos, se obtuvo que en los mejores casos para el algoritmo CD2S (Experimentos II y III), la media del tiempo de evaluación de CD2S fue 3,78 veces menor que la media de 2B2S, lo que indica que el tiempo de CD2S fue el 26 % del tiempo empleado por este algoritmo; 3,03 veces menor que IB2S y 2,85 veces menor que VC2S+. En el peor de los casos para el algoritmo CD2S (Experimento IV), el tiempo promedio del algoritmo CD2S fue menor en 1,22 veces y 1,19 veces que los algoritmos 2B2S y VC2S+ y, similar al tiempo del algoritmo IB2S. Además, en la mayoría de los lotes de simulaciones el algoritmo CD2S obtuvo el mejor tiempo individual.

Estos resultados se deben a las optimizaciones que realiza el algoritmo CD2S, en las comparaciones de dominancia por la separación del *skyline* en cuatro grupos, ya que en la mayoría de los casos, el algoritmo CD2S ejecutó el menor número de comparaciones de dominancia para obtener el *skyline*, con una diferencia de más de un orden de magnitud con respecto a las comparaciones de 2B2S e IB2S, y una diferencia menos marcada para el algoritmo VC2S+, aunque en algunos casos resultó seis veces menor que las comparaciones realizadas por este algoritmo.

El algoritmo VC2S+ resultó mejor en todos los experimentos, que los algoritmos 2B2S e IB2S, en la evaluación de consultas por transformaciones del ambiente por desplazamiento del punto de referencia, en particular en el Experimento III, porque al igual que CD2S actualiza el *skyline* en estas circunstancias. Incluso, en los casos en que no se disponía de dimensiones no espaciales, se acercaron un poco más los resultados a los del algoritmo CD2S. Sin embargo, observando el desempeño general del algoritmo, si se considera el tiempo

total (la suma de los tiempos empleados para el cambio de estatus y el desplazamiento del punto de referencia), éste resultó mayor que el tiempo del algoritmo IB2S. Esto se debe a que al ocurrir un cambio de estatus en los objetos, el algoritmo VC2S+ requiere el doble del tiempo del algoritmo IB2S, porque debe construir la nueva estructura con los objetos habilitados, generar el nuevo *skyline* no espacial (si existen preferencias no espaciales) y, visitar una mayor cantidad de objetos para evaluar el *skyline*.

El algoritmo IB2S, tuvo la mejor media en tiempo de evaluación en el Experimento IV para las variantes con la tasa de cambio igual a 70 % y 90 % y, la segunda mejor media en la mayoría de los experimentos. Esto se debe a la sencilla actualización del *skyline* que realiza en presencia de cambios de los objetos, de no ser así resultaría peor que el algoritmo VC2S+.

Los algoritmos que efectuaron menos comparaciones en el *skyline* fueron los algoritmos CD2S y VC2S+, siendo mejor el algoritmo CD2S, donde resultó una ventaja la exclusión de las verificaciones de pertenencia al *skyline*, de los objetos del *skyline* estático. Así como también, son los algoritmos que menos cambios en el *skyline* realizaron para obtener el conjunto.

Por los resultados anteriores, se concluye que el algoritmo CD2S en general tuvo un mejor desempeño, en tiempo y número de comparaciones.

Como trabajo futuro, se pretende simplificar u optimizar el mecanismo de actualización del *skyline* para el algoritmo CD2S en presencia de cambios en los objetos, para ofrecer una ventaja aún mayor.

Para tal objetivo, es posible una extensión del algoritmo, en donde luego de eliminar del *skyline* los objetos que ya no están habilitados, se calcule la región dominante de cada uno de los objetos candidatos, indicada en la Figura C.1(c), p. 145, y en caso de no ser vacía, se verifique si existe un objeto del *skyline* (con los objetos en **sEst** y **sDin**) que pertenezca a esa región. De ser así, el objeto automáticamente estaría dominado espacialmente y habría

que verificar si no es dominado en sus dimensiones no espaciales. Si esta región es vacía, el objeto no es dominado espacialmente. De esta manera, el número de comparaciones de dominancia puede ser reducido.

Esta región, se obtiene formando círculos centrados en cada punto de referencia y con un radio igual a la distancia a las coordenadas del candidato y, luego se realiza la intersección de estos círculos. Debido a que sólo un punto de referencia es móvil, se puede pre-calcular la intersección para los círculos correspondientes a los puntos estáticos y en el momento de la verificación, se efectúa la intersección con el círculo del punto móvil.

Además, se desea analizar el mecanismo de *Valid Scope* propuesto en [45], que define una región para la cual los resultados del *skyline* siguen siendo válidos y, así puede ser integrado con los patrones de cambio que emplean los algoritmos VC2S+ y CD2S (para determinar la región del *skyline* que cambió). De esta manera, se podría disminuir la cantidad de objetos comparados.

Se aspira estudiar un nuevo enfoque, combinando las consultas DSSQ con las consultas *skyline* direccionales (*directional skyline query*) [52]. Las cuales son relevantes en este trabajo de grado, porque ofrecen una posibilidad para disminuir los objetos del *skyline* que probablemente no sean interesantes para el usuario. Estas consultas, incluyen la dirección de movimiento del punto de referencia en el *skyline* y un ángulo máximo de desviación, los objetos que estén en otra dirección con respecto a ese punto, pueden ser descartados.

Otra ampliación al problema puede realizarse mediante el estudio del *skyline* espacial general (*general skyline*, GSSKY) [59]. De esta forma, se podrían diferenciar los puntos de referencia en distintos grupos, según su tipo y retornar los objetos cercanos a al menos un punto de cada tipo. Por ejemplo, según un mapa de una ciudad, un usuario desea conseguir la farmacia de turno más cercana y a la vez cercana a los cajeros automáticos de un determinado banco y de alguna estación de metro. La idea es retornar la farmacia de turno próxima a la ubicación del usuario y al menos a un punto de referencia del tipo cajero automático y un punto de referencia del tipo estación de metro.

Adicionalmente, es conveniente investigar la integración de este problema con técnicas de *ranking* como el *top-k* aplicados a objetos espaciales [67], para conseguir aquellos k-objetos interesantes a ser retornados en una primera fase del *skylines*.

En cuanto a las limitaciones de los algoritmos propuestos, éstos evalúan la consultas mediante la distancia Euclídea y considerando como atributo cambiante, un atributo que tiene un dominio de dos valores (*booleano*). Por lo que, estos algoritmos deben ser mejorados para emplear otras métricas de distancia más funcionales y admitir un dominio de valores mayor para el atributo altamente cambiante.

Por último, se espera que los algoritmos propuestos puedan ser integrados a sistemas que dispongan de sensores, para validar si un objeto en un determinado instante está habilitado. Esto permitiría, realizar pruebas en ambientes reales con la finalidad de fomentar el uso de los algoritmos implementados en situaciones de la vida cotidiana y probar su desempeño. También se desea probar los algoritmos, con dimensiones no espaciales correlacionadas a las dimensiones espaciales, ya que esto reduciría el espacio de búsqueda para los algoritmos VC2S+ y CD2S.

REFERENCIAS

- [1] M. Sharifzadeh, C. Shahabi, and L. Kazemi, “Processing spatial skyline queries in both vector spaces and spatial network databases,” in *ACM Transactions on Database Systems (TODS)*, vol. 34, (New York, NY, USA), pp. 14:1–14:45, September 2009.
- [2] P. Godfrey, R. Shipley, and J. Gryz, “Maximal vector computation in large data sets,” in *VLDB '05: Proceedings of the 31st international conference on Very large data bases*, pp. 229–240, 2005.
- [3] S. Borzsonyi, D. Kossmann, and K. Stocker, “The skyline operator,” in *Proceedings of the 17th International Conference on Data Engineering*, pp. 421–430, IEEE Computer Society, 2001.
- [4] H. T. Kung, F. Luccio, and F. P. Preparata, “On finding the maxima of a set of vectors,” *Journal of the ACM (JACM)*, vol. 22, pp. 469–476, October 1975.
- [5] K. Tan, P. Eng, and B. Ooi, “Efficient progressive skyline computation,” in *VLDB '01: Proceedings of the 27th International Conference on Very Large Data Bases*, (San Francisco, CA, USA), pp. 301–310, Morgan Kaufmann Publishers Inc., 2001.
- [6] S. Chaudhuri, N. Dalvi, and R. Kaushik, “Robust cardinality and cost estimation for skyline operator,” in *Data Engineering, 2006. ICDE '06. Proceedings of the 22nd International Conference on*, (Los Alamitos, CA, USA), p. 64, IEEE Computer Society, 2006.
- [7] M. Sharifzadeh and C. Shahabi, “The spatial skyline queries,” in *VLDB 2006: Proceedings of the 32nd international conference on Very large data bases*, pp. 751–762, VLDB Endowment, 2006.
- [8] K. Kodama, Y. Iijima, X. Guo, and Y. Ishikawa, “Skyline queries based on user locations and preferences for making location-based recommendations,” in *Proceedings of the 2009 International Workshop on Location Based Social Networks*, LBSN '09, (New York, NY, USA), pp. 9–16, ACM, 2009.
- [9] N. Lande and A. Lande, *The 10 Best of Everything, Second Edition: An Ultimate Guide for Travelers*. <http://travel.nationalgeographic.com/travel/top-10/sporting-events/#page=1>. National Geographic the 10 Best of Everything, National Geographic Society, 2008.

- [10] B. Yan-Zhong, S. Li-Min, Z. Hong-Song, Y. Ting-Xin, and L. Zheng-Jun, "A parking management system based on wireless sensor network," in *Acta Automatica Sinica* 32 (6), pp. 968–977, ISSN: 0254-4156, CN: 11-2109/TP, Institute of Software, Chinese Academy of Sciences, 2006.
- [11] I. Ganchev, M. O'Droma, and D. Meere, "An intelligent car park management system based on wireless sensor networks," in *Pervasive Computing and Applications*, 2006 1st International Symposium, 2006.
- [12] J. Benson, T. O'Donovan, P. O'Sullivan, U. Roedig, C. Sreenan, J. Barton, and B. Murphy, A. adn O'Flynn, "Car-park management using wireless sensor networks," in *Proceedings 2006 31st IEEE Conference on Local Computer Networks*, 2006.
- [13] S. Lee, D. Yoon, and A. Ghosh, "Intelligent parking lot application using wireless sensor networks," in *Collaborative Technologies and Systems, 2008. CTS 2008. International Symposium on*, 2008.
- [14] G. Yan, S. Olariu, M. C. Weigle, and M. Abueela, "Smartparking: A secure and intelligent parking system using notice," in *Intelligent Transportation Systems, 2008. ITSC 2008*, 11th International IEEE Conference on, 2008.
- [15] I. Ganchev, M. O'Droma, and D. Meere, "Intelligent car parking locator service," in *Int. Journal Information Technologies and Knowledge*, vol. 2, pp. 166–173, Institute of Information Theories and Applications FOI ITHEA, 2008.
- [16] E. Idris, M. and. Tamil, N. Noor, Z. Razak, and K. Fong, "Parking guidance system utilizing wireless sensor network and ultrasonic sensor," in *Information Technology Journal*, vol. 8, pp. 138–146, 2009.
- [17] D. Kossmann, F. Ramsak, and S. Rost, "Shooting stars in the sky: An online algorithm for skyline queries," in *VLDB '02 Proceedings of the 28th international conference on Very Large Data Bases*, pp. 275–286, 2002.
- [18] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "An optimal and progressive algorithm for skyline queries," in *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, (New York, NY, USA), pp. 467–478, ACM Press, 2003.
- [19] M. Kokkodis and P. Bhattacharya, "Implementation of skyline query algorithms," in *University of California Riverside*, 2005.
- [20] Chen, Lei, Lian, and Xiang, "Dynamic skyline queries in metric spaces," in *Proceedings of the 11th international conference on Extending database technology, EDBT '08*, (New York, NY, USA), pp. 333–343, ACM, 2008.
- [21] M. M. Astrahan, M. W. Blasgen, D. D. Chamberlin, K. P. Eswaran, J. N. Gray, P. P. Griffiths, W. F. King, R. A. Lorie, P. R. McJones, J. W. Mehl, G. R. Putzolu, I. L. Traiger, B. W. Wade, and V. Watson, "System R: relational approach to database management," *ACM Trans. Database Syst.*, vol. 1, no. 2, pp. 97–137, 1976.

- [22] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, "Skyline with presorting," in *Data Engineering, 2003. Proceedings. 19th International Conference on Data Engineering (ICDE 2003)*, 2003.
- [23] C. yee Chong, Ieee, S. P. Kumar, and S. Member, "Sensor networks: evolution, opportunities, and challenges," in *Proceedings of the IEEE, Vol. 91, No. 8, August 2003*, vol. 91, pp. 1247–1256, 2003.
- [24] J. Lee and S.-w. Hwang, "Skytree: scalable skyline computation for sensor data," in *SensorKDD '09: Proceedings of the Third International Workshop on Knowledge Discovery from Sensor Data*, (New York, NY, USA), pp. 114–123, ACM, 2009.
- [25] H. Shen, Z. Chen, and X. Deng, "Location-based skyline queries in wireless sensor networks," in *NSWCTC '09: Proceedings of the 2009 International Conference on Networks Security, Wireless Communications and Trusted Computing*, (Washington, DC, USA), pp. 391–395, IEEE Computer Society, 2009.
- [26] Z. Zhang, R. Cheng, D. Papadias, and A. K. Tung, "Minimizing the communication cost for continuous skyline maintenance," in *Proceedings of the 35th SIGMOD international conference on Management of data*, SIGMOD '09, (New York, NY, USA), pp. 495–508, ACM, 2009.
- [27] S. Yoon and C. Shahabi, "Poster abstract: A distributed algorithm to compute spatial skyline in wireless sensor networks," in *IPSN '09: Proceedings of the 2009 International Conference on Information Processing in Sensor Networks*, (Washington, DC, USA), pp. 375–376, IEEE Computer Society, 2009.
- [28] B. Chen and W. Liang, "Progressive skyline query processing in wireless sensor networks," in *MSN '09: Proceedings of the 2009 Fifth International Conference on Mobile Ad-hoc and Sensor Networks*, (Washington, DC, USA), pp. 17–24, IEEE Computer Society, 2009.
- [29] B. Chen, W. Liang, and J. X. Yu, "Progressive skyline query evaluation and maintenance in wireless sensor networks," in *CIKM '09: Proceeding of the 18th ACM conference on Information and knowledge management*, (New York, NY, USA), pp. 1445–1448, ACM, 2009.
- [30] P. Godfrey, "Cardinality estimation of skyline queries," Tech. Rep. CS-2002-03, York University, Department of Computer Science, 2002.
- [31] F. Di Bartolo, M. Goncalves, I. Martínez, and F. Sardá, "An evolutionary algorithm for skyline-join optimization," in *Vol. pp. 276 - 287, 13th Portuguese Conference in Artificial Intelligence - EPIA 2007*. Guimaraes, Portugal, 2007.
- [32] K. Lee, B. Zheng, H. Li, and W. Lee, "Approaching the skyline in z order," in *Proceedings of the 33rd international conference on Very large data bases, VLDB'07*, pp. 279–290, 2007.

- [33] F. Di Bartolo, F. Sardá, M. Goncalves, and I. Martínez, “Una extensión al optimizador de postgresql para consultas skyline,” in *Vol. 4, pp. 70 - 82.*, (Venezuela), Revista Faraute de Ciencias y Tecnología. Indexada en el LATINDEX CATÁLOGO, 2009.
- [34] F. Di Bartolo, M. Goncalves, and F. Sardá, “Una propuesta de modelo de costo para consultas skyline,” in *CD. pp. 1 - 10*, VI Workshop Chileno de Bases de Datos de las Jornadas Chilenas de Computación 2008. Punta Arenas, Chile, 2008.
- [35] D. Papadias, Y. Tao, G. Fu, and B. Seeger, “Progressive skyline computation in database systems,” in *ACM Transactions on Database Systems (TODS) - Special Issue: SIGMOD/PODS 2003*, vol. 30, (New York, NY, USA), pp. 41–82, 2005.
- [36] W. Balke, U. Guntzer, and J. Zheng, “Efficient distributed skylining for web information systems,” in *Advances in Database Technology - EDBT 2004*, vol. 2992, pp. 256–273, Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2004.
- [37] X. Lin, Y. Yuan, W. Wang, and H. Lu, “Stabbing the sky: Efficient skyline computation over sliding windows,” in *ICDE '05: Proceedings of the 21st International Conference on Data Engineering*, (Washington, DC, USA), pp. 502–513, IEEE Computer Society, 2005.
- [38] P. Wu, C. Zhang, Y. Feng, B. Y. Zhao, D. Agrawal, and A. E. Abbadi, “Parallelizing skyline queries for scalable distribution,” in *Advances in Database Technology - EDBT 2006*, pp. 112–130, 2006.
- [39] L. Chen, B. Cui, H. Lu, L. Xu, and Q. Xu, “isky: Efficient and progressive skyline computing in a structured p2p network,” in *Distributed Computing Systems, International Conference on*, vol. 0, (Los Alamitos, CA, USA), pp. 160–167, IEEE Computer Society, 2008.
- [40] S. Wang, Q. H. Vu, B. C. Ooi, A. K. Tung, and L. Xu, “Skyframe: a framework for skyline query processing in peer-to-peer systems,” in *The VLDB Journal*, vol. 18, (Secaucus, NJ, USA), pp. 345–362, Springer-Verlag New York, Inc., 2009.
- [41] W. Son, M.-W. Lee, H.-K. Ahn, and S.-W. Hwang, “Spatial skyline queries: An efficient geometric algorithm,” in *Proceedings of the 11th International Symposium on Advances in Spatial and Temporal Databases*, SSTD '09, (Berlin, Heidelberg), pp. 247–264, Springer-Verlag, 2009.
- [42] M.-W. Lee, W. Son, H.-K. Ahn, and S.-W. Hwang, “Spatial skyline queries: exact and approximation algorithms,” *Geoinformatica*, vol. 15, pp. 665–697, Oct. 2011.
- [43] K. Deng, X. Zhou, and H. Tao, “Multi-source skyline query processing in road networks,” *Data Engineering, International Conference on*, vol. 0, pp. 796–805, 2007.
- [44] C. Li, A. K. H. Tung, W. Jin, and M. Ester, “On dominating your neighborhood profitably,” in *VLDB '07: Proceedings of the 33rd international conference on Very large data bases*, pp. 818–829, VLDB Endowment, 2007.

- [45] B. Zheng, K. Lee, and W.-C. Lee, “Location-dependent skyline query,” in *Mobile Data Management, 2008. MDM ’08. 9th International Conference on*, pp. 148 –155, april 2008.
- [46] D. H. Qiang Pu, Ahmed Lbath, “Location based recommendation for mobile users using language model and skyline query,” in *International Journal of Information Technology and Computer Science*, 2012.
- [47] X. Guo, Y. Ishikawa, and Y. Gao, “Direction-based spatial skylines,” in *Proceedings of the Ninth ACM International Workshop on Data Engineering for Wireless and Mobile Access, MobiDE ’10*, (New York, NY, USA), pp. 73–80, ACM, 2010.
- [48] X. Guo, B. Zheng, Y. Ishikawa, and Y. Gao, “Direction-based surround queries for mobile recommendations,” *The VLDB Journal*, vol. 20, pp. 743–766, Oct. 2011.
- [49] K. B. P. Iyer and V. Shanthi, “Goal directed relative skyline queries in time dependent road networks,” *CoRR*, vol. abs/1205.1853, 2012.
- [50] K. P. Iyer and V. Shanthi, “Spatial boolean skyline boundary queries in road networks,” in *Computing Communication Networking Technologies (ICCCNT), 2012 Third International Conference on*, pp. 1 –6, july 2012.
- [51] E. El-Dawy, H. Mokhtar, and A. El-Bastawissy, “Multi-level continuous skyline queries (mcsq),” in *Data and Knowledge Engineering (ICDKE), 2011 International Conference on*, pp. 36 –40, sept. 2011.
- [52] E. El-Dawy, H. Mokhtar, and A. El-Bastawissy, “Directional skyline queries,” in *Data and Knowledge Engineering* (Y. Xiang, M. Pathan, X. Tao, and H. Wang, eds.), vol. 7696 of *Lecture Notes in Computer Science*, pp. 15–28, Springer Berlin Heidelberg, 2012.
- [53] Y.-K. Huang, C.-H. Chang, and C. Lee, “Continuous distance-based skyline queries in road networks,” *Information Systems*, vol. 37, no. 7, pp. 611 – 633, 2012.
- [54] J. Lim, Y. Park, K. Bok, and J. Yoo, “A continuous reverse skyline query processing considering the mobility of query objects,” in *Internet and Distributed Computing Systems* (Y. Xiang, M. Pathan, X. Tao, and H. Wang, eds.), vol. 7646 of *Lecture Notes in Computer Science*, pp. 227–237, Springer Berlin Heidelberg, 2012.
- [55] J. Lim, Y. Park, K. Bok, and J. Yoo, “An efficient continuous reverse skyline query processing method over moving objects,” in *Ubiquitous Information Technologies and Applications* (Y.-H. Han, D.-S. Park, W. Jia, and S.-S. Yeo, eds.), vol. 214 of *Lecture Notes in Electrical Engineering*, pp. 323–331, Springer Netherlands, 2013.
- [56] W. Son, S.-w. Hwang, and H.-K. Ahn, “Mssq: Manhattan spatial skyline queries,” in *Advances in Spatial and Temporal Databases* (D. Pfoser, Y. Tao, K. Mouratidis, M. Nascimento, M. Mokbel, S. Shekhar, and Y. Huang, eds.), vol. 6849 of *Lecture Notes in Computer Science*, pp. 313–329, Springer Berlin Heidelberg, 2011.

- [57] T. Skopal and J. Lokoc, “Answering metric skyline queries by pm-tree.,” in *DATESO* (J. Pokorný, V. Snásel, and K. Richta, eds.), vol. 567 of *CEUR Workshop Proceedings*, pp. 22–37, CEUR-WS.org, 2010.
- [58] N. M. Soudani and A. Baraani-Dastgerdi, “The spatial nearest neighbor skyline queries,” *CoRR*, vol. abs/1112.2336, 2011.
- [59] Q. Lin, Y. Zhang, W. Zhang, and A. Li, “General spatial skyline operator,” in *Proceedings of the 17th international conference on Database Systems for Advanced Applications - Volume Part I*, DASFAA’12, (Berlin, Heidelberg), pp. 494–508, Springer-Verlag, 2012.
- [60] Q. Lin, Y. Zhang, W. Zhang, and X. Lin, “Efficient general spatial skyline computation,” *World Wide Web*, pp. 1–24, 2012.
- [61] G. won You, M.-W. Lee, H. Im, and S. won Hwang, “The farthest spatial skyline queries,” *Information Systems*, vol. 38, no. 3, pp. 286 – 301, 2013.
- [62] N. Roussopoulos, S. Kelley, and F. Vincent, “Nearest neighbor queries,” *SIGMOD Rec.*, vol. 24, pp. 71–79, May 1995.
- [63] G. R. Hjaltason and H. Samet, “Distance browsing in spatial databases,” *ACM Trans. Database Syst.*, vol. 24, pp. 265–318, June 1999.
- [64] D. Papadias, Y. Tao, K. Mouratidis, and C. K. Hui, “Aggregate nearest neighbor queries in spatial databases,” *ACM Trans. Database Syst.*, vol. 30, pp. 529–576, June 2005.
- [65] M. E. Ali, E. Tanin, R. Zhang, and R. Kotagiri, “Probabilistic voronoi diagrams for probabilistic moving nearest neighbor queries,” *Data & Knowledge Engineering*, vol. 75, no. 0, pp. 1 – 33, 2012.
- [66] F. Wenzel, M. Endres, S. Mandl, and W. Kiessling, “Complex preference queries supporting spatial applications for user groups,” *Proc. VLDB Endow.*, vol. 5, pp. 1946–1949, Aug. 2012.
- [67] W. Liu, Y. Jing, K. Chen, and W. Sun, “Combining top-k query in road networks,” in *Proceedings of the 2011 international conference on Web-Age Information Management*, WAIM’11, (Berlin, Heidelberg), pp. 63–75, Springer-Verlag, 2012.
- [68] J. a. B. Rocha-Junior and K. Nørvåg, “Top-k spatial keyword queries on road networks,” in *Proceedings of the 15th International Conference on Extending Database Technology*, EDBT ’12, (New York, NY, USA), pp. 168–179, ACM, 2012.
- [69] D. Yates, E. Nahum, J. Kurose, and P. Shenoy, “Data quality and query cost in wireless sensor networks,” in *PERCOMW ’07: Proceedings of the Fifth IEEE International Conference on Pervasive Computing and Communications Workshops*, pp. 272–278, IEEE Computer Society, 2007.

- [70] Y. Kwon, J.-H. Choi, Y.-D. Chung, and S. Lee, “In-network processing for skyline queries in sensor networks,” in *IEICE transactions on communications*, Oxford University Press, 2007.
- [71] H. Chen, S. Zhou, and J. Guan, “Towards energy-efficient skyline monitoring in wireless sensor networks,” in *EWSN’07: Proceedings of the 4th European conference on Wireless sensor networks*, (Berlin, Heidelberg), pp. 101–116, Springer-Verlag, 2007.
- [72] J. Xin, G. Wang, L. Chen, X. Zhang, and Z. Wang, “Continuously maintaining sliding window skylines in a sensor network,” in *DASFAA’07: Proceedings of the 12th international conference on Database systems for advanced applications*, (Berlin, Heidelberg), pp. 509–521, Springer-Verlag, 2007.
- [73] J. Huang, J. Xin, G. Wang, and M. Li, “Efficient k-dominant skyline processing in wireless sensor networks,” in *9th International Conference on Hybrid Intelligent Systems (HIS 2009), August 12-14, 2009, Shenyang, China*, pp. 289–294, IEEE Computer Society, 2009.
- [74] W. Liang, B. Chen, and J. X. Yu, “Energy-efficient skyline query processing and maintenance in sensor networks,” in *CIKM ’08: Proceeding of the 17th ACM conference on Information and knowledge management*, (New York, NY, USA), pp. 1471–1472, ACM, 2008.
- [75] K. Pripužić, H. Belani, and M. Vuković, “Early forest fire detection with sensor networks: Sliding window skylines approach,” in *KES ’08: Proceedings of the 12th international conference on Knowledge-Based Intelligent Information and Engineering Systems, Part I*, (Berlin, Heidelberg), pp. 725–732, Springer-Verlag, 2008.
- [76] S. Kim and S. O. Yang, “Query processing algorithm in wireless sensor networks,” in *International Journal of Multimedia and Ubiquitous Engineering*, 2009.
- [77] C. Yu, C. Lu, and S. Kuo, “Secure multidimensional queries in tiered sensor networks,” *CoRR*, vol. abs/0911.4238, 2009.
- [78] Z. Huang, C. S. Jensen, H. Lu, and B. C. Ooi, “Skyline queries against mobile lightweight devices in manets,” in *ICDE ’06: Proceedings of the 22nd International Conference on Data Engineering*, (Washington, DC, USA), p. 66, IEEE Computer Society, 2006.
- [79] Y. Yang, S. Papadopoulos, D. Papadias, and G. Kollios, “Spatial outsourcing for location-based services,” in *ICDE ’08: Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, (Washington, DC, USA), pp. 1082–1091, IEEE Computer Society, 2008.
- [80] N. Chen, L. Shou, G. Chen, Y. Gao, and J. Dong, “Predictive skyline queries for moving objects,” in *Proceedings of the 14th International Conference on Database Systems for Advanced Applications*, DASFAA ’09, (Berlin, Heidelberg), pp. 278–282, Springer-Verlag, 2009.

- [81] N. Chen, L.-d. Shou, G. Chen, Y.-j. Gao, and J.-x. Dong, “Prismo: predictive skyline query processing over moving objects,” *Journal of Zhejiang University SCIENCE C*, vol. 13, pp. 99–117, 2012.
- [82] W. Lee, C. S.-H. Eom, and T.-C. Jo, “Path skyline for moving objects,” in *Proceedings of the 14th Asia-Pacific international conference on Web Technologies and Applications*, APWeb’12, (Berlin, Heidelberg), pp. 610–617, Springer-Verlag, 2012.
- [83] Y. Ge, H. Xiong, A. Tuzhilin, K. Xiao, M. Gruteser, and M. Pazzani, “An energy-efficient mobile recommender system,” in *KDD ’10: Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, (New York, NY, USA), pp. 899–908, ACM, 2010.
- [84] C. Yoo, S. Kim, and D. Pahng, “Parking guidance and management system,” in *US_Patent, US 6107942 A*, Premier Management Partners, Inc., 2000.
- [85] N. Katenka, E. Levin, and G. Michailidis, “Robust target localization from binary decisions in wireless sensor networks,” in *Technometrics: A journal of statistics for the physical, chemical and engineering sciences*, vol. 50, pp. 448–461, Department of Statistics, The University of Michigan, 2008.
- [86] G. Voronoi, “Nouvelles applications des paramètres continus à la théorie des formes quadratiques, Recherches sur les Parallelloedres Primitifs,” in *Journal fur die Reine und Angewandte Mathematik*, vol. 134, pp. 198–287, 1908.
- [87] B. Delaunay, “Sur la sphère vide. A la mémoire de Georges Voronoi,” in *Izvestia Akademii Nauk SSSR, Otdelenie Matematicheskikh i Estestvennykh Nauk (Bulletin of Academy of Sciences of the USSR)*, vol. 7, pp. 793–800, 1934.
- [88] B. Hafenrichter and W. Kießling, “Optimization of relational preference queries,” in *ADC ’05: Proceedings of the 16th Australasian database conference*, pp. 175–184, 2005.

APÉNDICE A

ARCHIVOS DE ENTRADA

Los datos de entrada de los distintos algoritmos implementados para su inicialización, se transfieren a través de los archivos: `aConf`, `aPtoRef` y `aDimNoEsp`. Una instancia de estos archivos, es lo que se denomina un caso de prueba.

A continuación se realiza un ejemplo con cada uno de los archivos para explicar el contenido de los mismos.

Archivo `aConf`

Este archivo se utiliza para definir el tamaño del plano y la disponibilidad de las localidades en él. Se supone que una localidad mide 1×1 de la unidad u^2 (Ej: m^2) y no existe separación entre localidades.

```
3 - 8 - 101111110010110111111111
```

Figura A.1: Ejemplo del archivo `aConf`

El ejemplo de la Figura A.1, muestra que el plano es de 3 filas por 8 columnas, por lo que tiene 24 localidades u objetos. Se marca con 1 los puestos libres y con 0 los ocupados.

Archivo `aPtoRef`

Este archivo permite definir la cantidad de puntos de referencias, sus posiciones e identificar el punto de referencia móvil. Cabe destacar que las coordenadas son relativas al plano, siendo el punto (0,0) la esquina superior izquierda del plano.

2,6 - 1,1 - 1,2 - 1,6 2

Figura A.2: Ejemplo del archivo aPtoRef

El ejemplo de la Figura A.2, establece que existen 4 puntos de referencia, cuya posición es la siguiente: $p_0 = (2, 6)$, $p_1 = (1, 1)$, $p_2 = (1, 2)$ y $p_3 = (1, 6)$ y además que p_2 es el punto de referencia móvil.

Archivo aDimNoEsp

Este archivo se utiliza si los objetos tienen atributos no espaciales, define la cantidad de dimensiones y los valores de estas para cada uno de los objetos en el plano.

2-
0.487539230392,4.19351255707
26.2381945724,26.3347885924
23.4403171748,9.27142875584
27.7368414551,14.8034415519
29.2885448938,21.7049367942
4.08180573385,22.0823880608
20.7161378618,28.348194663
28.2949134887,19.7640552512
18.2739834282,4.77926890261
29.3382414266,17.3399332021
25.9823297084,1.07466038332
19.4177523378,2.82174713353
27.3977870164,29.9634764768
2.75292603915,8.26276404556
6.04723101654,13.3003200389
29.1611579196,19.4479598268
25.6498214104,26.9408202116
13.1587126762,24.6948368649
23.310814192,6.98534610226
17.8195695547,4.20699909846
10.7591372506,21.4898811532
17.3900852952,5.72196282036
7.70356222599,8.88541782281
18.0056280043,4.35442624616

Figura A.3: Ejemplo del archivo aDimNoEsp

El ejemplo de la Figura A.3, indica que existen 2 dimensiones no espaciales y provee los valores de estas dimensiones para cada una de las 24 localidades. Cada fila del archivo

corresponde a una localidad.

Ejemplo de un objeto

Los objetos son enumerados como las casillas de un tablero, por lo que se toma como objeto 0, e_0 que tiene la posición $(0, 0)$ del plano, sucesivamente hasta el objeto 23, e_{23} que tiene la posición $(2, 7)$.

Estos serían los atributos del objeto 0, según los archivos de entrada:

- * **Posición:** $(0, 0)$.
- * **Estatus:** disponible.
- * **Dimensiones espaciales:** $d_0 = dist(e_0, p_0)$, $d_1 = dist(e_0, p_1)$, $d_2 = dist(e_0, p_2)$ y $d_3 = dist(e_0, p_3)$. Donde $dist$ es la distancia euclíadiana de la posición del objeto e_0 a la posición del punto de referencia p_i .
- * **Dimensiones no espaciales:** $d_4 = 0,487539230392$ y $d_5 = 4,19351255707$.

APÉNDICE B

SIMULADOR

A continuación, se muestra en el Algoritmo 13, el detalle del Simulador empleado en el estudio experimental:

Algoritmo 13 Simulador

```

1: procedure MAIN(aConf, aPtosRef, aDimNoEsp, n, modo, %Cambio, distMax)
2:   for i  $\leftarrow$  1, n do
3:     if i = 1 then                                > Modo inicial
4:       2B2S.COMENZAR(aConf, aPtosRef, aDimNoEsp)
5:       IB2S.COMENZAR(aConf, aPtosRef, aDimNoEsp)
6:       VC2S+.COMENZAR(aConf, aPtosRef, aDimNoEsp)
7:       CD2S.COMENZAR(aConf, aPtosRef, aDimNoEsp)
8:       CONOCERAMBIENTEINICIAL(aConf, aPtosRef, aDimNoEsp)
9:     else
10:      tipo  $\leftarrow$  DETERMINARTIPOCAMBIO(modo)
11:      if tipo = 1 then                          > Cambio de disponibilidad
12:        ocupados, disponibles  $\leftarrow$  ALTERARDISPONIBILIDAD(%Cambio)
13:        2B2S.CALCULARSOLUCIONCAMBIOESTATUS(ocupados, disponibles)
14:        IB2S.CALCULARSOLUCIONCAMBIOESTATUS(ocupados, disponibles)
15:        VC2S+.CALCULARSOLUCIONCAMBIOESTATUS(ocupados, disponibles)
16:        CD2S.CALCULARSOLUCIONCAMBIOESTATUS(ocupados, disponibles)
17:      else if tipo = 2 then                    > Desplazamiento
18:        pos  $\leftarrow$  ALTERARPUNTOREFERENCIA(distMax)
19:        2B2S.CALCULARSOLUCIONDESPLAZAMIENTO(pos)
20:        IB2S.CALCULARSOLUCIONDESPLAZAMIENTO(pos)
21:        VC2S+.CALCULARSOLUCIONDESPLAZAMIENTO(pos)
22:        CD2S.CALCULARSOLUCIONDESPLAZAMIENTO(pos)
23:      end if
24:    end if
25:  end for
26: end procedure

```

El Simulador, en la primera iteración ejecuta el procedimiento COMENZAR de cada uno de los algoritmos con el ambiente inicial y lo almacena para luego hacer las modificaciones (Líneas 3-9). En las siguientes iteraciones, el simulador determina el tipo de cambio a realizar según el modo permitido (Línea 10).

Si el tipo de cambio corresponde al cambio de disponibilidad (Líneas 11-16), se realiza la transformación según el parámetro `%Cambio` en la Línea 12 y se ejecuta el procedimiento `calcularSolucionCambioEstatus` de cada uno de los algoritmos.

Por el contrario, si se escoge el desplazamiento del punto de referencia (Líneas 17-23), se realiza la transformación según el parámetro `DistMax` en la Línea 18 y se ejecuta el procedimiento `calcularSolucionDesplazamiento` de cada uno de los algoritmos.

APÉNDICE C

CREACIÓN DE LA REGIÓN DE BÚSQUEDA

Originalmente, en [1], la región de búsqueda de la Figura C.1(a) se generaba con cada objeto en `skylineNoEsp` (Ej: s_1), tomando cada punto de referencia (Ej: q_1, q_2, q_3) se crea un círculo centrado en ese punto con radio la distancia del punto de referencia al objeto candidato ($C_i(q_1, s_1), C_i(q_2, s_1), C_i(q_3, s_1)$). En la Figura C.1(c), la región amarilla fuera de los círculos, contiene los objetos dominados por ese punto *skyline* s_1 (en la figura es denotado como p), denominada región de dominancia de p ; en la región blanca dentro de estos, están los objetos que dominan a p en cercanía al punto de referencia centro del círculo en el que se encuentran, pero no necesariamente dominan a p en el resto de las dimensiones. Si la intersección de todos los círculos no es vacía, se le denomina región dominante de p , y los objetos dentro de esa región son los que lo dominan espacialmente. En este caso como p se encuentra dentro del *convex hull* esa intersección es vacía.

Posteriormente, se unen los círculos y el mínimo rectángulo de la región resultante formará la región de visibilidad del *convex hull* con respecto al objeto *skyline* en cuestión ($regionBusqueda(s_1)$). En cada iteración, se unen las regiones de visibilidad de esta manera, $regionBusqueda(s_1) \cup regionBusqueda(s_2)$, formando la región de búsqueda hasta esa iteración, la cual será la máxima región donde podrá encontrarse un objeto *skyline* espacial. Los objetos que se encuentren fuera de esta región, son irrelevantes, por lo cual serán descartados.

Para simplificar los costos de procesamiento, se realiza un proceso equivalente en la Figura C.1(b). Se generan en lugar de círculos, cuadrados centrados en el punto de referencia ($C_u(q_1, s_1), C_u(q_2, s_1), C_u(q_3, s_1)$) y cuyo largo es dos veces la distancia desde ese punto al

objeto *skyline*. Por lo tanto, la tarea de obtener el rectángulo de búsqueda se realiza con el mínimo rectángulo que abarca la unión de estas regiones formadas por los cuadrados. Nótese que para crear este mínimo rectángulo, basta con saber las distancias a s_1 y la posición de s_1 y, con esos valores se toman el mínimo y el máximo punto que debe cubrir el mínimo rectángulo tanto en el eje x como en el eje y.

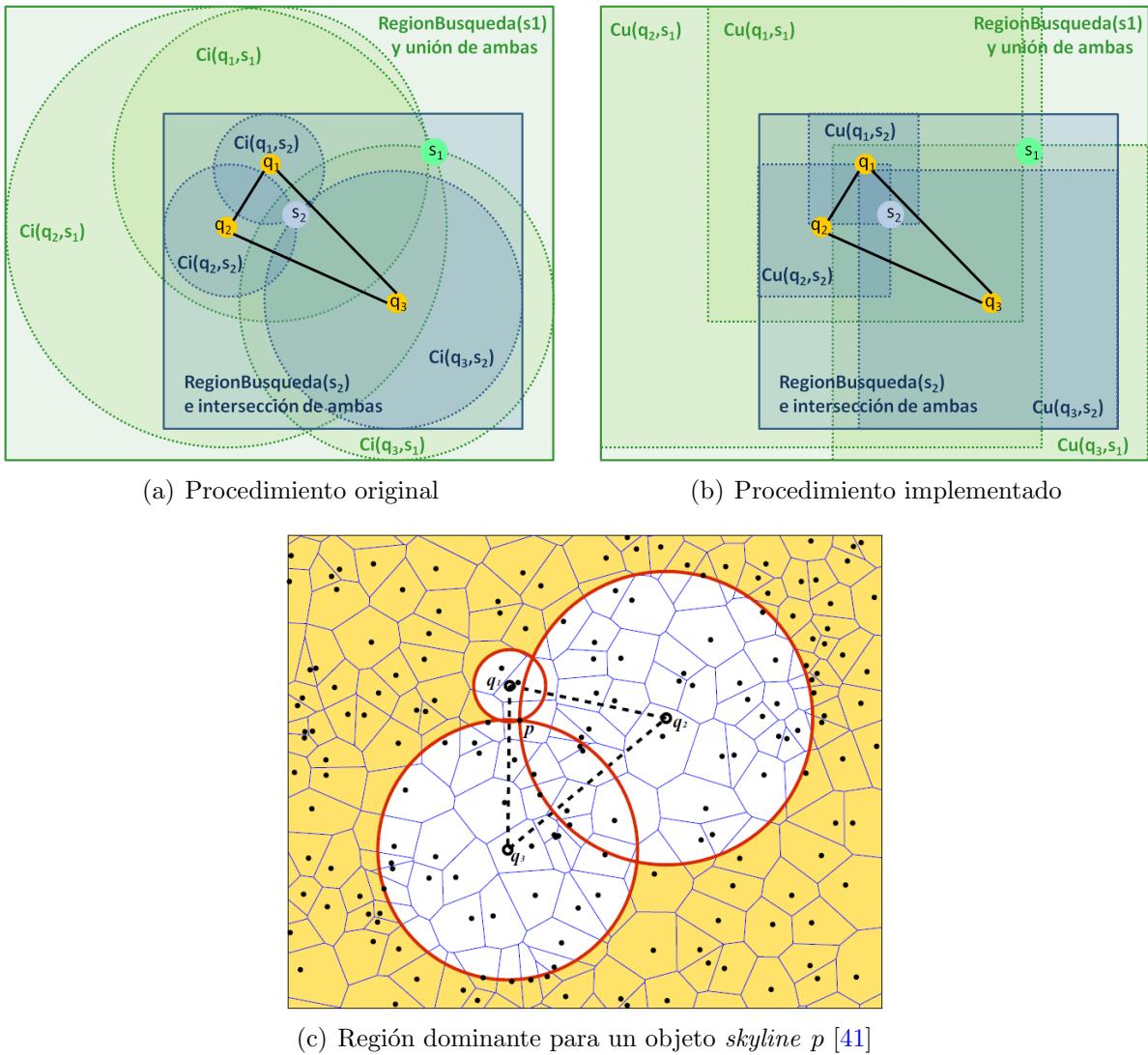


Figura C.1: Ejemplo de regiones de búsqueda

APÉNDICE D

CARACTERÍSTICAS DE LOS ALGORITMOS

Representación del plano	
2B2S IB2S	Crea una matriz de <i>bytes</i> colocando <i>bits</i> en 1 para los objetos disponibles y 0 los ocupados.
VC2S+	Crea un grafo <i>Delaunay</i> para almacenar sólo los objetos disponibles.
Rango de exploración de candidatos	
2B2S IB2S	El plano completo.
VC2S+	Se determina mediante una heurística, si se tienen únicamente dimensiones espaciales. Se suspende la exploración de la rama al ser dominada la celda Voronoi del objeto. Si existen dimensiones no espaciales, se crea una región de búsqueda con el <i>skyline</i> no espacial. Para el caso de desplazamiento, se excluyen los objetos dentro de la región sin cambios, la cual define el sector que no altera al <i>skyline</i> .
CD2S	Está dado por la región de búsqueda tanto para dimensiones no espaciales como espaciales (aun cuando se calculan de forma distinta). Se almacenan como candidatos los objetos disponibles que estén dentro de la región de búsqueda y fuera del <i>convex hull</i> . Los objetos fuera de la region de búsqueda, se descartan de inmediato y si además pertenecían al <i>skyline</i> (en una ejecución anterior), se eliminan del mismo.
Búsqueda de candidatos (teniendo sólo dimensiones espaciales)	
2B2S IB2S	Explora cada <i>bit</i> de la matriz buscando sólo los objetos disponibles para agregarlos al índice de candidatos.
VC2S+	Atraviesa el grafo partiendo del objeto más cercano al punto de referencia móvil. Explora cada vecino del objeto de partida y sucesivamente cada vecino a su vez. Ordena la pila de exploración por la suma de las dimensiones en forma ascendente y poda la búsqueda en el momento que la celda Voronoi del objeto es dominada por el <i>skyline</i> y la pila de posibles <i>skyline</i> .
CD2S	Similar a 2B2S, pero sólo se revisan los objetos en la matriz que estén dentro de la región de búsqueda. El plano puede ser explorado tomando mas de un <i>byte</i> a la vez hasta un <i>long</i> y solo aquellas regiones distintas de 0 son revisadas para conseguir los candidatos. Los objetos que estén dentro del <i>convex hull</i> son agregados directamente al <i>skyline</i> y al indice correspondiente del <i>convex hull</i> , con ellos actualiza la región de búsqueda para afinarla.

Tabla D.1: Cuadro comparativo de los algoritmos (1 de 4)

Búsqueda de candidatos (teniendo dimensiones no espaciales)	
2B2S IB2S	No hay diferencia al caso de sólo dimensiones espaciales.
VC2S+	Atraviesa el grafo de la misma forma, pero mientras cada rama de exploración esté dentro de la región de búsqueda formada por el <i>skyline</i> no espacial, se sigue ampliando y diversificando. Se corta la rama en el momento que el objeto esté fuera de la región de búsqueda.
CD2S	Similar al caso de sólo dimensiones espaciales, pero la región de búsqueda formada por el <i>skyline</i> no espacial sólo se actualiza con esos objetos. En caso de ser la primera vez que se ejecute el algoritmo, se debe explorar el plano completo para encontrar todos los objetos <i>skyline</i> no espaciales.
Composición del <i>skyline</i>	
2B2S IB2S	No hay distinción entre los objetos del <i>skyline</i> .
VC2S+	3 partes: <i>convex hull</i> , espacial y no espacial. En caso de desplazamiento, no cambia la parte no espacial.
CD2S	4 partes: <i>convex hull</i> estático, <i>convex hull</i> dinámico, espacial y no espacial. En caso de desplazamiento, no cambian las partes: <i>convex hull</i> estático y no espacial.
Construcción inicial del <i>skyline</i> espacial	
2B2S IB2S	Llevada a cabo una vez encontrados los candidatos, ejecuta el BNL <i>skyline</i> para todos los candidatos considerando dimensiones espaciales y no espaciales.
VC2S+	Realizada a medida que se atraviesa el grafo. Si encuentra un objeto del <i>convex hull</i> lo agrega al <i>skyline</i> sin verificar dominancia (estos objetos ya pueden ser retornados si se necesitara), sino, evalúa (considerando todas las dimensiones) si su celda Voronoi es dominada contra el <i>skyline</i> y la pila de posibles <i>skyline</i> , si no lo es, lo agrega a la pila de candidatos. Al ser escogido para la exploración de sus vecinos, se realiza otra evaluación de dominancia contra el <i>skyline</i> y la pila de posibles <i>skyline</i> , de no ser dominado, lo agrega la pila. Al final de la exploración del grafo, se ejecuta otra verificación por cada objeto en la pila de posibles <i>skyline</i> . Para un mismo objeto se podría realizar hasta 3 verificaciones antes de colocarlo como <i>skyline</i> , apartando las comparaciones que se realizan estando del lado de la ventana al estar en la pila de posibles <i>skyline</i> .
CD2S	Inicialmente encuentra los candidatos dentro del <i>convex hull</i> para agregarlos directamente al <i>skyline</i> , al igual que VC2S+, estos ya pueden ser retornados pero los separa según la porción del <i>skyline</i> en la que estén. Luego calcula el <i>skyline</i> de los candidatos que están fuera del <i>convex hull</i> y dentro de la región de búsqueda. Se supone que un candidato no puede dominar a objetos del <i>convex hull</i> , sólo podrán ser incomparables, por lo que si se encuentra que el candidato es mejor en alguna dimensión, no se revisan el resto de las dimensiones. En la ventana están de primeros los objetos pertenecientes al <i>convex hull</i> ya que son los más aptos para descartar a posibles candidatos <i>skyline</i> en menos iteraciones. Sólo se verifica la dominancia de los objetos que están dentro de la región de búsqueda y fuera del <i>convex hull</i> .

Tabla D.2: Cuadro comparativo de los algoritmos (2 de 4)

Construcción inicial del <i>skyline</i> no espacial	
2B2S IB2S	No hay distinción, se realiza de la forma anterior.
VC2S+ CD2S	Se deben iterar todos los objetos disponibles considerando sólo las dimensiones no espaciales para obtener el <i>skyline</i> no espacial, éste formará parte de la ventana <i>skyline</i> al realizar las posteriores verificaciones. El <i>skyline</i> no espacial no se alterará, se puede retornar si es necesario.
Comparaciones de dominancia	
2B2S IB2S	Compara cada dimensión del objeto contra la del objeto de la ventana, identificando si es mejor, peor o igual. Si en todas las dimensiones el objeto es mejor o igual al de la ventana lo elimina del <i>skyline</i> , si es peor o igual lo descartar o si es incomparable, lo mantiene y continua con las comparaciones del siguiente objeto de la ventana.
VC2S+	Para obtener el <i>skyline</i> no espacial, se realizan las mismas comparaciones que en 2B2S, pero sólo con las dimensiones no espaciales. Para el <i>skyline</i> espacial, compara cada dimensión (espacial y no espacial) del objeto contra la del objeto de la ventana, verificando si no es dominado, al momento en que el objeto sea mejor en alguna dimensión, suspende las comparaciones por dimensiones y pasa a la evaluación contra el siguiente objeto de la ventana para determinar si es dominado o incomparable.
CD2S	Para el <i>skyline</i> no espacial, se realizan las mismas comparaciones que en VC2S+. En el espacial, varía según el caso, si el objeto de la ventana pertenece al <i>convex hull</i> , el candidato no dominará al de la ventana, así que si en alguna dimensión espacial domina, se suspenden las comparaciones y se asume incomparable para esa iteración. Sólo se recurre a la comparación de las dimensiones no espaciales, si con las dimensiones espaciales no se pudo determinar dominancia. En el caso 3 de desplazamiento, como sólo se itera el <i>skyline</i> para eliminar los objetos que ya no deberían estar, únicamente verifica si el objeto continua siendo mejor en la dimensión del punto móvil, si es así, sigue siendo incomparable. Si en algún momento llegase a ser peor o igual en esa dimensión, entonces se verifican en el resto de las dimensiones si el objeto en al menos una es mejor, en ese caso se suspende la comparación porque sigue siendo incomparable. Si no domina en alguna, entonces es descartado y por lo tanto eliminado del <i>skyline</i> . Para el cambio de estatus, en el momento en que se actualiza el <i>skyline</i> espacial, se realizan comparaciones verificando sólo estas dimensiones.

Tabla D.3: Cuadro comparativo de los algoritmos (3 de 4)

Actualización del <i>skyline</i>	
2B2S	No ocurre.
IB2S	Sólo se realiza en el <i>modo actualización de estatus</i> .
VC2S+	Sólo se realiza en el <i>modo actualización de desplazamiento</i> .
CD2S	Ocurre en ambos casos: <i>modo actualización de desplazamiento</i> y <i>modo actualización de estatus</i> (en este último actualiza también la porción no espacial).
Modo actualización de estatus	
2B2S	Actualiza la disponibilidad en el plano, reinicia los candidatos y el <i>skyline</i> y ejecuta la solución inicial.
IB2S	Actualiza la disponibilidad en el plano, elimina de candidatos y del <i>skyline</i> los objetos no disponibles. Si no se modificó el <i>skyline</i> , sólo se verifica la dominancia contra el <i>skyline</i> para los nuevos objetos disponibles. Sino, toma todos los candidatos.
VC2S+	Actualiza los candidatos según los cambios ocurridos, con ellos construye de nuevo el grafo, crea el nuevo <i>skyline</i> no espacial y ejecuta desde el inicio el VS^2 para obtener el nuevo <i>skyline</i> .
CD2S	Similar al IB2S con la diferencia que actualiza el <i>skyline</i> no espacial y utiliza las propiedades del <i>convex hull</i> y la región de búsqueda para obtener los candidatos faltantes y así actualizar el <i>skyline</i> espacial.
Modo actualización de desplazamiento	
2B2S IB2S	Actualiza las distancias de los candidatos al punto móvil y calcula de nuevo el <i>skyline</i> .
VC2S+	Actualiza las distancias de los candidatos al punto móvil, crea el nuevo <i>convex hull</i> , clasifica el caso según uno de los 6 patrones, para el patrón 1 devuelve el mismo <i>skyline</i> , para el 6 ejecuta desde el inicio el VS^2 y para el 2,3,4 y 5 calcula la región sin cambios para excluir los objetos que no necesitan ser revisados y actualiza el <i>skyline</i> ejecutando el VCS^2 . No es necesario actualizar ni recalcular el <i>skyline</i> no espacial, ya que se mantiene.
CD2S	Realiza la misma clasificación que VC2S+ aunque para el patrón 6 también actualiza el <i>skyline</i> . En los casos 2,3,4 y 5 el <i>skyline</i> espacial es actualizado iterando los índices del <i>convex hull</i> dinámico y del <i>skyline</i> con la ayuda de la nueva región de búsqueda para ubicar los candidatos faltantes (si lo amerita), excluye a los pertenecientes a la región sin cambios. No es necesario actualizar, ni recalcular el <i>skyline</i> no espacial, ni revisar los objetos en el índice del <i>convex hull</i> estático, ya que pudo variar el <i>convex hull</i> completo, pero el formado por los puntos estáticos se mantiene.

Tabla D.4: Cuadro comparativo de los algoritmos (4 de 4)

APÉNDICE E

TABLAS DE RESULTADOS

A continuación algunos resultados de las principales métricas e indicadores utilizadas para realizar el estudio experimental, las cuales son: `tiempo promedio`, `mejor tiempo`, `peor tiempo`, `vecesTmin`, `vecesTmax` y `#comparaciones de dominancia` para cada algoritmo en cada combinación y la cardinalidad del `skyline`. Estos resultados corresponden a los datos recolectados por lotes de 50 simulaciones por cada algoritmo en los casos de prueba construidos. Con cada caso de prueba se realizaron 10 lotes de simulaciones.

Los resultados del Experimento I pueden ser vistos en las Tablas E.1 y E.2 para las variantes 1 y 2 con la configuración 1. Los resultados del Experimento II pueden ser vistos en las Tablas E.3 y E.4 para las variantes 1 y 2 con la configuración 2 y las Tablas E.15 y E.16 para las variantes 3 y 4 con la configuración 3.

Los resultados del Experimento III que se basa en el movimiento del punto de referencia, pueden ser vistos en las Tablas E.5, E.6, E.7, E.8 y E.9 para las variantes 1, 2, 3, 4 y 5 con la configuración 2. Los resultados del Experimento IV que se basa en el cambio de disponibilidad, pueden ser vistos en la Tablas E.10, E.11, E.12, E.13 y E.14 para las variantes 1, 2, 3, 4 y 5 con la configuración 2.

Los resultados del Experimento V que se basa en una cardinalidad media de objetos, pueden ser vistos en la Tabla E.17 para la variante 1 la configuración 4, la Tabla E.18 para la variante 2 la configuración 5 y la Tabla E.19 para la variante 3 la configuración 6. Los resultados del Experimento VI que se basa en una cardinalidad alta de objetos, pueden ser vistos en la Tabla E.20 para la variante 1 con la configuración 7.

Configuración: 1. Se utilizan 4 casos de prueba equivalentes por combinación.

Variante 1. Modo: combinado; %Cambio: aleatorio; DistMax: 10

Algoritmo	Combinación											
	1	2	3	4	5	6	7	8	9	10	11	12
Tiempo promedio (ms)												
2B2S	49316	110892	169084	360575	563949	878152	798539	920735	496550	772299	1542451	1708400
IB2S	41783	90937	147721	322162	503923	817485	728131	837361	466988	696692	1461021	1563869
VC2S+	65089	89282	170694	378672	508140	898906	921838	880228	555589	915575	1811409	1745067
CD2S	24013	46536	92722	223446	295346	458056	470483	494990	284148	389203	968175	937090
Mejor tiempo (ms)												
2B2S	8723	41736	72914	54815	136481	124609	527708	539288	172381	473661	582217	790977
IB2S	8040	30212	60154	50814	108673	86904	41506	465602	161964	435061	513459	685748
VC2S+	10622	28190	72702	75491	103458	75801	49009	489945	179127	523324	482635	710200
CD2S	3822	14364	35453	36967	48149	34538	24983	286416	102828	205765	253286	457913
Peor tiempo (ms)												
2B2S	189603	217406	301570	820403	1329638	1925376	1901760	1426212	1229000	1406230	3013111	2466277
IB2S	162124	181119	263182	720271	1147348	1842028	1740514	1365961	112903	1327684	2928675	2339452
VC2S+	273093	179095	353244	700218	1539370	2040525	2389492	1474580	1027848	1409246	3717239	2993878
CD2S	95477	92480	157095	420322	729211	1056118	1304416	859583	504559	623772	2153662	1463652
vecesTMin (lotes)												
2B2S	0	0	0	0	0	0	0	0	0	0	0	0
IB2S	0	0	0	0	0	0	0	0	0	0	0	0
VC2S+	0	0	0	0	0	0	0	0	0	0	0	0
CD2S	40	40	40	40	40	40	40	40	40	40	40	40
vecesTMax (lotes)												
2B2S	2	31	18	10	32	19	15	23	12	12	13	19
IB2S	0	0	0	0	0	0	0	1	0	0	0	0
VC2S+	38	9	22	30	8	21	25	16	28	28	27	21
CD2S	0	0	0	0	0	0	0	0	0	0	0	0
#Comparaciones de dominancia												
2B2S	0.66×10^9	1.34×10^9	1.83×10^9	2.97×10^9	3.84×10^9	5.10×10^9	5.23×10^9	6.39×10^9	4.99×10^9	7.12×10^9	10.83×10^9	12.44×10^9
IB2S	0.52×10^9	1.06×10^9	1.49×10^9	2.44×10^9	3.14×10^9	4.20×10^9	4.30×10^9	5.27×10^9	4.10×10^9	5.83×10^9	8.99×10^9	10.24×10^9
VC2S+	0.19×10^9	0.40×10^9	0.71×10^9	1.62×10^9	0.63×10^9	1.15×10^9	1.55×10^9	2.24×10^9	0.83×10^9	1.50×10^9	1.99×10^9	3.02×10^9
CD2S	0.07×10^9	0.17×10^9	0.32×10^9	0.70×10^9	0.24×10^9	0.32×10^9	0.53×10^9	0.77×10^9	0.26×10^9	0.42×10^9	0.64×10^9	1.03×10^9
Skyline	55101	79590	94408	121759	143317	160539	155600	170053	144559	164133	219860	216541

Tabla E.1: Resultados por lotes para la configuración 1 (combinado-aleatorio-10)

Configuración: 1. Se utilizan 4 casos de prueba equivalentes por combinación.

Variante 2. Modo: combinado; %Cambio: aleatorio; DistMax: 20

Algoritmo	Combinación											
	1	2	3	4	5	6	7	8	9	10	11	12
Tiempo promedio (ms)												
2B2S	63826	131434	182779	355573	534866	928414	756383	938349	483424	791650	1467206	1666322
IB2S	52123	109861	154926	322695	463911	857086	686551	849108	445346	718996	1414663	1533423
VC2S+	67546	120559	177115	379644	461547	942160	867192	974519	540421	941276	1740645	1720130
CD2S	31734	62044	100956	228182	269460	487179	436841	543839	286516	409366	922453	953248
Mejor tiempo (ms)												
2B2S	10884	32791	46769	57378	134255	127923	53258	428786	194880	494703	762130	855317
IB2S	9481	27410	39510	50362	116231	93797	44354	384964	170419	458322	722818	709781
VC2S+	16021	23158	53099	56324	120145	79649	50801	345811	200899	458386	643662	581262
CD2S	5030	15652	25332	27972	59111	34422	29059	187043	119826	223232	386543	377762
Pior tiempo (ms)												
2B2S	247606	339113	587423	773537	1330004	1921101	1636041	1899559	829445	1312052	2858450	2528677
IB2S	193660	297576	488268	721761	1110762	1930779	1460508	1867227	837846	1202424	2793496	2286595
VC2S+	168747	375354	553109	867596	1045928	2529491	2033117	1974670	154293	1733596	3249326	2835853
CD2S	100187	150461	280113	479044	508269	1183792	1183784	1071122	654048	741922	1875256	1657588
vecesTMin (lotes)												
2B2S	0	0	0	0	0	0	0	0	0	0	0	0
IB2S	1	0	0	1	0	0	0	0	0	0	0	0
VC2S+	0	0	0	0	0	0	0	0	0	0	0	0
CD2S	39	40	40	39	40	40	40	40	40	40	40	40
vecesTMax (lotes)												
2B2S	10	24	18	16	30	23	12	14	22	7	7	16
IB2S	0	0	0	0	0	1	0	0	0	0	0	0
VC2S+	30	16	22	24	10	16	28	26	18	33	33	24
CD2S	0	0	0	0	0	0	0	0	0	0	0	0
#Comparaciones de dominancia												
2B2S	0.79×10 ⁹	1.44×10 ⁹	1.86×10 ⁹	2.98×10 ⁹	3.70×10 ⁹	5.33×10 ⁹	5.06×10 ⁹	6.39×10 ⁹	4.90×10 ⁹	7.20×10 ⁹	10.62×10 ⁹	12.19×10 ⁹
IB2S	0.62×10 ⁹	1.15×10 ⁹	1.49×10 ⁹	2.44×10 ⁹	2.98×10 ⁹	4.41×10 ⁹	4.13×10 ⁹	5.26×10 ⁹	4.04×10 ⁹	5.88×10 ⁹	8.91×10 ⁹	10.02×10 ⁹
VC2S+	0.25×10 ⁹	0.46×10 ⁹	0.70×10 ⁹	1.55×10 ⁹	0.59×10 ⁹	1.23×10 ⁹	1.57×10 ⁹	2.34×10 ⁹	0.86×10 ⁹	1.48×10 ⁹	1.89×10 ⁹	2.92×10 ⁹
CD2S	0.10×10 ⁹	0.19×10 ⁹	0.31×10 ⁹	0.67×10 ⁹	0.22×10 ⁹	0.36×10 ⁹	0.54×10 ⁹	0.80×10 ⁹	0.28×10 ⁹	0.42×10 ⁹	0.61×10 ⁹	1.04×10 ⁹
Skyline	61342	83058	94530	121488	138420	166503	153026	169486	143896	164709	218585	214186

Tabla E.2: Resultados por lotes para la configuración 1 (combinado-aleatorio-20)

Configuración: 2. Se utilizan 4 casos de prueba equivalentes por combinación.

Variante 1. Modo: combinado; % Cambio: aleatorio; DistMax: 10

Algoritmo					Combinación							
	1	2	3	4	5	6	7	8	9	10	11	12
Tiempo promedio (ms)												
2B2S	1285	1985	6197	11279	2982	3827	6582	13889	6015	10244	16180	23286
IB2S	1662	2120	5429	9366	2941	3507	5613	11352	5150	8169	13089	18481
VC2S+	2528	7159	11015	12200	3423	8250	10149	14034	4853	8934	14883	16756
CD2S	337	2568	4262	5914	705	2984	3954	7066	1132	4038	6855	8914
Mejor tiempo (ms)												
2B2S	117	490	1093	5095	444	1380	1820	4273	1221	3750	3819	8646
IB2S	567	953	1514	4309	947	1727	2005	3704	1700	3430	3538	7136
VC2S+	1228	4114	5428	5961	1504	4568	4354	4187	2267	3911	4630	7398
CD2S	37	1678	2111	3443	149	1865	2033	2699	186	2404	2700	3813
Pior tiempo (ms)												
2B2S	6360	4564	16929	34218	9612	10290	22320	35165	18359	58900	61993	38476
IB2S	5179	4215	13757	28967	7951	8757	18652	27344	14450	37104	46500	33155
VC2S+	5564	10366	22026	29201	7222	13110	22695	27483	12091	27540	47458	27140
CD2S	1553	4066	8283	16103	2827	4935	9567	15035	3888	14333	27140	14190
vecesTMin (lotes)												
2B2S	0	30	10	0	0	11	2	0	0	0	0	0
IB2S	0	3	0	0	0	4	6	0	0	0	0	0
VC2S+	0	0	0	0	0	0	0	0	0	0	0	0
CD2S	40	7	30	40	40	25	32	40	40	40	40	40
vecesTMax (lotes)												
2B2S	2	0	0	0	14	8	0	3	16	29	25	35
IB2S	0	0	0	0	1	0	0	0	0	0	0	0
VC2S+	38	40	40	26	31	40	37	24	11	15	21	5
CD2S	0	0	0	0	0	0	0	0	0	0	0	0
#Comparaciones de dominancia												
2B2S	0.66x10 ⁸	1.19x10 ⁸	3.17x10 ⁸	6.09x10 ⁸	1.72x10 ⁸	2.36x10 ⁸	4.30x10 ⁸	8.24x10 ⁸	3.90x10 ⁸	5.94x10 ⁸	9.09x10 ⁸	13.79x10 ⁸
IB2S	0.48x10 ⁸	0.85x10 ⁸	2.34x10 ⁸	4.36x10 ⁸	1.29x10 ⁸	1.73x10 ⁸	3.13x10 ⁸	5.93x10 ⁸	2.90x10 ⁸	4.39x10 ⁸	6.78x10 ⁸	10.14x10 ⁸
VC2S+	0.13x10 ⁸	0.27x10 ⁸	0.92x10 ⁸	1.61x10 ⁸	0.28x10 ⁸	0.45x10 ⁸	0.91x10 ⁸	2.04x10 ⁸	0.34x10 ⁸	0.74x10 ⁸	1.60x10 ⁸	2.42x10 ⁸
CD2S	0.06x10 ⁸	0.21x10 ⁸	0.64x10 ⁸	1.15x10 ⁸	0.14x10 ⁸	0.31x10 ⁸	0.62x10 ⁸	1.54x10 ⁸	0.16x10 ⁸	0.46x10 ⁸	1.08x10 ⁸	1.80x10 ⁸
Skyline	11618	14714	23867	32395	17982	19391	24547	33816	22693	27676	34785	40253

Tabla E.3: Resultados por lotes para la configuración 2 (combinado-aleatorio-10)

Configuración: 2. Se utilizan 4 casos de prueba equivalentes por combinación.
Variante 2. Modo: combinado; %Cambio: aleatorio; DistMax: 20

Algoritmo					Combinación							
	1	2	3	4	5	6	7	8	9	10	11	12
Tiempo promedio (ms)												
2B2S	3273	4256	9271	20441	5592	6770	16020	22354	11191	11171	20531	38058
IB2S	3026	3762	7619	16805	4802	5620	12627	17698	8870	8914	16439	29100
VC2S+	4231	8887	13155	19617	5493	9970	15629	20491	8216	9876	17179	24551
CD2S	1026	3368	5552	10139	1577	4144	7553	11217	2656	4782	8785	14498
Mejor tiempo (ms)												
2B2S	315	814	1902	6435	965	2268	2506	6242	1837	3868	5252	10999
IB2S	755	1186	2143	5083	1446	2395	2655	5265	1949	3463	4851	8895
VC2S+	1741	4543	6089	7948	2131	4647	8681	7169	2766	6464	7974	8632
CD2S	137	1812	2769	4094	315	2237	3212	3685	359	2583	3610	5847
Pior tiempo (ms)												
2B2S	13117	23524	30979	84993	15922	25868	121940	53922	76897	38531	65114	145594
IB2S	9185	18487	22979	73061	11729	18517	89698	42857	59899	27514	48515	108528
VC2S+	9642	21626	30838	52442	12477	18387	57145	48447	39965	20679	52066	61307
CD2S	2903	8633	17152	30067	4793	8583	36374	26399	17585	11285	30559	38453
vecesTMin (lotes)												
2B2S	0	13	5	0	0	2	1	0	0	0	0	0
IB2S	0	9	3	0	0	4	3	0	0	0	0	0
VC2S+	0	0	0	0	0	0	0	0	0	0	0	0
CD2S	40	18	32	40	40	34	36	40	40	40	40	40
vecesTMax (lotes)												
2B2S	5	1	3	15	16	4	10	22	30	25	28	38
IB2S	0	0	0	0	0	0	0	0	0	0	0	0
VC2S+	35	39	37	25	24	36	30	18	10	15	12	2
CD2S	0	0	0	0	0	0	0	0	0	0	0	0
#Comparaciones de dominancia												
2B2S	1.30×10 ⁸	1.96×10 ⁸	3.96×10 ⁸	7.79×10 ⁸	2.67×10 ⁸	3.50×10 ⁸	6.68×10 ⁸	10.28×10 ⁸	5.66×10 ⁸	6.39×10 ⁸	10.56×10 ⁸	17.06×10 ⁸
IB2S	0.95×10 ⁸	1.44×10 ⁸	2.92×10 ⁸	5.85×10 ⁸	2.00×10 ⁸	4.96×10 ⁸	7.51×10 ⁸	4.21×10 ⁸	4.73×10 ⁸	7.99×10 ⁸	12.70×10 ⁸	
VC2S+	0.33×10 ⁸	0.48×10 ⁸	1.22×10 ⁸	2.57×10 ⁸	0.48×10 ⁸	0.67×10 ⁸	1.67×10 ⁸	2.94×10 ⁸	0.59×10 ⁸	0.85×10 ⁸	1.95×10 ⁸	3.37×10 ⁸
CD2S	0.18×10 ⁸	0.35×10 ⁸	0.87×10 ⁸	1.73×10 ⁸	0.26×10 ⁸	0.48×10 ⁸	1.11×10 ⁸	2.13×10 ⁸	0.32×10 ⁸	0.56×10 ⁸	1.36×10 ⁸	2.52×10 ⁸
Skyline	16599	19006	27389	39485	21889	23592	32693	40760	26827	29558	38678	46764

Tabla E.4: Resultados por lotes para la configuración 2 (combinado-aleatorio-20)

Configuración: 2. Se utilizan 4 casos de prueba equivalentes por combinación.

Variante 1. Modo: actualización movimiento; %Cambio: no aplica; DistMax: 5

Algoritmo	Combinación											
	1	2	3	4	5	6	7	8	9	10	11	12
Tiempo promedio (ms)												
2B2S	2579	4647	11780	28986	8229	10683	15179	32800	14373	24162	40012	60437
IB2S	2595	4635	11789	29024	8238	10684	15181	32782	14353	24194	40017	60659
VC2S+	1582	14530	24085	31045	3668	20311	24270	32175	4330	17173	30127	35940
CD2S	750	3991	7156	10582	3039	7320	7355	9619	3273	7664	11672	11557
Mejor tiempo (ms)												
2B2S	187	886	2470	8456	850	3188	2993	8881	3336	8455	9597	17970
IB2S	125	884	2495	8395	810	3101	2965	8811	3269	8564	9592	18081
VC2S+	374	3028	10526	2120	450	5929	6443	3666	344	654	8611	10981
CD2S	93	388	2151	600	120	1467	1967	1030	140	249	2017	2527
Peor tiempo (ms)												
2B2S	8021	17110	59949	103410	35472	36504	57751	92258	35261	83819	120373	209896
IB2S	8044	17083	59578	103230	35549	36365	57037	92244	35282	84149	120560	213848
VC2S+	6125	30373	55037	89730	13513	44219	46895	59104	13366	37795	73257	100774
CD2S	3847	13856	26911	38170	16304	32548	26321	23747	12949	22193	38356	59843
vecesTMin (lotes)												
2B2S	0	4	5	0	0	0	3	3	0	0	0	0
IB2S	0	9	4	0	0	0	3	2	0	0	0	0
VC2S+	0	0	0	0	6	0	0	0	5	0	0	0
CD2S	40	27	31	40	34	34	35	40	35	40	40	40
vecesTMax (lotes)												
2B2S	6	0	1	12	17	1	2	9	23	13	16	10
IB2S	15	0	0	3	21	0	4	6	17	16	11	27
VC2S+	19	40	39	25	2	39	34	25	0	11	13	3
CD2S	0	0	0	0	0	0	0	0	0	0	0	0
#Comparaciones de dominancia												
2B2S	0.12×10 ⁹	0.22×10 ⁹	0.53×10 ⁹	1.15×10 ⁹	0.37×10 ⁹	0.49×10 ⁹	0.78×10 ⁹	1.51×10 ⁹	0.73×10 ⁹	1.11×10 ⁹	1.72×10 ⁹	2.63×10 ⁹
IB2S	0.12×10 ⁹	0.22×10 ⁹	0.53×10 ⁹	1.15×10 ⁹	0.37×10 ⁹	0.49×10 ⁹	0.78×10 ⁹	1.51×10 ⁹	0.73×10 ⁹	1.11×10 ⁹	1.72×10 ⁹	2.63×10 ⁹
VC2S+	0.02×10 ⁹	0.05×10 ⁹	0.13×10 ⁹	0.25×10 ⁹	0.04×10 ⁹	0.08×10 ⁹	0.14×10 ⁹	0.30×10 ⁹	0.04×10 ⁹	0.10×10 ⁹	0.23×10 ⁹	0.33×10 ⁹
CD2S	0.01×10 ⁹	0.05×10 ⁹	0.09×10 ⁹	0.14×10 ⁹	0.05×10 ⁹	0.10×10 ⁹	0.16×10 ⁹	0.03×10 ⁹	0.09×10 ⁹	0.17×10 ⁹	0.18×10 ⁹	
Skyline	15109	20604	29086	43716	25581	26920	32424	44852	30809	38187	46595	52638

Tabla E.5: Resultados por lotes para la configuración 2 (movimiento-N/A-5). Los algoritmos 2B2S y IB2S tienen valores similares porque no varían en este modo de ejecución.

Configuración: 2. Se utilizan 4 casos de prueba equivalentes por combinación.

Variante 2. Modo: actualización movimiento; %Cambio: no aplica; DistMax: 10

Algoritmo	Combinación											
	1	2	3	4	5	6	7	8	9	10	11	12
Tiempo promedio (ms)												
2B2S	9425	7353	25270	51048	14522	14822	35.84	59942	27777	40621	60092	81444
IB2S	9414	7378	25283	51025	14506	14798	35098	60040	27856	40680	60247	81454
VC2S+	5691	17186	34804	48175	9114	23291	3502	52789	12551	26810	44999	49948
CD2S	3813	5431	13043	16728	7473	9402	14179	20910	10439	17018	21111	18153
Mejor tiempo (ms)												
2B2S	672	1639	3812	13216	3621	3162	4697	11772	3386	9374	10481	21575
IB2S	660	1556	3801	13228	3523	3097	4682	11754	3368	9220	10529	21665
VC2S+	1093	4494	13982	19191	1968	10248	14656	14662	815	11480	13105	18646
CD2S	298	750	3708	4087	839	2104	3136	4325	294	2561	2873	5599
Pior tiempo (ms)												
2B2S	96021	20124	129027	222178	46034	52303	149018	203865	103956	175732	251222	186375
IB2S	96063	20061	129698	222251	45941	52217	148991	205857	103587	176700	253333	187076
VC2S+	41619	34586	120525	128529	34293	47361	82109	136811	49168	71574	186063	124080
CD2S	33556	16192	61622	47704	32308	30530	49183	68730	35883	57659	103284	62059
vecesTMin (lotes)												
2B2S	1	2	2	0	0	1	3	0	0	0	0	0
IB2S	0	2	1	0	0	1	1	0	0	0	0	0
VC2S+	0	0	0	0	4	0	0	0	5	0	0	0
CD2S	39	36	37	40	36	38	36	40	35	40	40	40
vecesTMax (lotes)												
2B2S	15	0	2	9	17	2	8	14	12	11	9	14
IB2S	9	0	3	7	22	0	5	9	27	19	21	22
VC2S+	16	40	35	24	1	38	27	17	0	10	10	4
CD2S	0	0	0	0	0	0	0	0	1	0	0	0
#Comparaciones de dominancia												
2B2S	0.24×10 ⁹	0.30×10 ⁹	0.73×10 ⁹	1.40×10 ⁹	0.51×10 ⁹	0.59×10 ⁹	1.12×10 ⁹	1.88×10 ⁹	1.06×10 ⁹	1.45×10 ⁹	2.03×10 ⁹	2.91×10 ⁹
IB2S	0.24×10 ⁹	0.30×10 ⁹	0.73×10 ⁹	1.40×10 ⁹	0.51×10 ⁹	0.59×10 ⁹	1.12×10 ⁹	1.88×10 ⁹	1.06×10 ⁹	1.45×10 ⁹	2.03×10 ⁹	2.91×10 ⁹
VC2S+	0.05×10 ⁹	0.08×10 ⁹	0.22×10 ⁹	0.40×10 ⁹	0.08×10 ⁹	0.12×10 ⁹	0.24×10 ⁹	0.48×10 ⁹	0.09×10 ⁹	0.17×10 ⁹	0.34×10 ⁹	0.44×10 ⁹
CD2S	0.06×10 ⁹	0.08×10 ⁹	0.17×10 ⁹	0.24×10 ⁹	0.09×10 ⁹	0.13×10 ⁹	0.19×10 ⁹	0.32×10 ⁹	0.09×10 ⁹	0.18×10 ⁹	0.27×10 ⁹	0.27×10 ⁹
Skyline	20810	23716	35860	50784	31334	30859	38916	53098	37543	43444	52628	57528

Tabla E.6: Resultados por lotes para la configuración 2 (movimiento-N/A-10). Los algoritmos 2B2S y IB2S tienen valores similares porque no varian en este modo de ejecución.

Configuración: 2. Se utilizan 4 casos de prueba equivalentes por combinación.

Variante 3. Modo: actualización movimiento; %Cambio: no aplica; DistMax: 15

Algoritmo	Combinación											
	1	2	3	4	5	6	7	8	9	10	11	12
Tiempo promedio (ms)												
2B2S	9188	13557	39275	57471	24592	19292	56476	106584	45598	55913	73354	120803
IB2S	9180	13552	39256	57486	24469	19292	56391	106623	45639	55888	73264	120832
VC2S+	7151	21334	40329	51394	14510	25633	46665	76951	21160	36928	51740	71205
CD2S	4652	8446	16508	19458	12169	11473	21294	33284	18727	23955	28481	30327
Mejor tiempo (ms)												
2B2S	581	1454	3428	13396	2664	3357	5618	19764	5023	10406	17208	24742
IB2S	623	1426	3463	13294	2581	3460	5491	19795	5025	10342	17410	24728
VC2S+	1170	8968	14822	23030	2216	9721	11049	30576	2477	13277	24588	26207
CD2S	356	1721	4991	6768	1382	2979	4400	9245	1282	4679	11481	6864
Pior tiempo (ms)												
2B2S	33265	49511	281190	129743	133992	82526	266422	525194	274791	235358	201240	531427
IB2S	33208	49889	282488	130028	133052	82396	267725	526294	275400	234658	201430	527422
VC2S+	18304	34818	137041	103304	57276	76563	149697	348490	100656	96751	117420	204813
CD2S	12013	19188	70375	48315	73200	45867	74551	157543	125529	78999	64168	114429
vecesTMin (lotes)												
2B2S	0	1	4	0	0	1	0	0	0	0	0	0
IB2S	0	4	0	0	0	2	1	0	0	0	0	0
VC2S+	0	0	0	0	5	0	0	0	6	0	0	0
CD2S	40	35	36	40	35	37	39	40	34	40	40	40
vecesTMax (lotes)												
2B2S	13	0	6	10	20	3	10	16	19	17	18	16
IB2S	13	2	3	13	18	2	12	9	21	13	14	22
VC2S+	14	38	31	17	2	35	18	15	0	10	8	2
CD2S	0	0	0	0	0	0	0	0	0	0	0	0
#Comparaciones de dominancia												
2B2S	0.28×10 ⁹	0.44×10 ⁹	0.91×10 ⁹	1.50×10 ⁹	0.68×10 ⁹	1.49×10 ⁹	2.42×10 ⁹	1.35×10 ⁹	1.76×10 ⁹	2.35×10 ⁹	3.47×10 ⁹	
IB2S	0.28×10 ⁹	0.44×10 ⁹	0.91×10 ⁹	1.50×10 ⁹	0.68×10 ⁹	1.49×10 ⁹	2.42×10 ⁹	1.35×10 ⁹	1.76×10 ⁹	2.35×10 ⁹	3.47×10 ⁹	
VC2S+	0.07×10 ⁹	0.13×10 ⁹	0.27×10 ⁹	0.46×10 ⁹	0.12×10 ⁹	0.15×10 ⁹	0.34×10 ⁹	0.64×10 ⁹	0.14×10 ⁹	0.23×10 ⁹	0.41×10 ⁹	0.62×10 ⁹
CD2S	0.07×10 ⁹	0.13×10 ⁹	0.22×10 ⁹	0.28×10 ⁹	0.13×10 ⁹	0.15×10 ⁹	0.28×10 ⁹	0.44×10 ⁹	0.14×10 ⁹	0.24×10 ⁹	0.34×10 ⁹	0.41×10 ⁹
Skyline	23456	27852	39750	53681	33622	32398	46007	61213	41376	47954	57007	65357

Tabla E.7: Resultados por lotes para la configuración 2 (movimiento-N/A-15). Los algoritmos 2B2S y IB2S tienen valores similares porque no varian en este modo de ejecución.

Configuración: 2. Se utilizan 4 casos de prueba equivalentes por combinación.

Variante 4. Modo: actualización movimiento; %Cambio: no aplica; DistMax:20

Algoritmo	Combinación											
	1	2	3	4	5	6	7	8	9	10	11	12
Tiempo promedio (ms)												
2B2S	13408	17797	46788	90077	37082	38176	50653	108258	53129	75280	82159	139724
IB2S	13402	17805	46818	90073	36980	38175	50654	108565	53197	75257	82134	140036
VC2S+	10511	24793	48582	72523	22628	34322	44098	83043	37841	44264	54542	85636
CD2S	6845	10382	21079	32631	17453	18960	21815	37912	24810	33557	32212	42420
Mejor tiempo (ms)												
2B2S	732	2091	5041	18989	2964	4840	9419	12698	7420	14317	11765	37160
IB2S	842	1966	5005	18861	3057	4772	9378	12741	7523	14354	11790	37675
VC2S+	1375	10298	16910	26516	2838	11840	18625	22075	5528	13119	18001	28425
CD2S	483	2538	5883	8653	1390	4272	6538	5995	2307	7709	6819	13185
Peor tiempo (ms)												
2B2S	77938	72606	158528	388890	224455	266291	250785	287321	254572	327818	339458	594782
IB2S	77765	72351	158011	387853	224499	265930	248212	287356	253191	329783	344574	601929
VC2S+	37960	50791	118136	193766	99515	116658	105863	164404	137219	139065	172281	261937
CD2S	27467	28348	51824	118841	79885	84229	72945	89997	113161	109091	95849	167763
vecesTMin (lotes)												
2B2S	0	1	4	0	0	0	0	0	0	0	0	0
IB2S	0	3	3	0	0	0	1	0	0	0	0	0
VC2S+	0	0	0	0	2	0	0	0	1	3	0	0
CD2S	40	36	33	40	38	40	39	40	39	37	40	40
vecesTMax (lotes)												
2B2S	16	1	5	12	17	7	7	9	15	9	17	21
IB2S	8	2	6	15	19	3	12	18	18	21	14	19
VC2S+	16	37	29	13	4	30	21	13	7	10	9	0
CD2S	0	0	0	0	0	0	0	0	0	0	0	0
#Comparaciones de dominancia												
2B2S	0.35×10 ⁹	0.51×10 ⁹	1.03×10 ⁹	1.80×10 ⁹	0.87×10 ⁹	0.99×10 ⁹	1.44×10 ⁹	2.50×10 ⁹	1.54×10 ⁹	2.05×10 ⁹	2.41×10 ⁹	3.70×10 ⁹
IB2S	0.35×10 ⁹	0.51×10 ⁹	1.03×10 ⁹	1.80×10 ⁹	0.87×10 ⁹	0.99×10 ⁹	1.44×10 ⁹	2.50×10 ⁹	1.54×10 ⁹	2.05×10 ⁹	2.41×10 ⁹	3.70×10 ⁹
VC2S+	0.10×10 ⁹	0.15×10 ⁹	0.34×10 ⁹	0.63×10 ⁹	0.16×10 ⁹	0.20×10 ⁹	0.34×10 ⁹	0.73×10 ⁹	0.19×10 ⁹	0.27×10 ⁹	0.41×10 ⁹	0.70×10 ⁹
CD2S	0.10×10 ⁹	0.15×10 ⁹	0.27×10 ⁹	0.40×10 ⁹	0.117×10 ⁹	0.21×10 ⁹	0.28×10 ⁹	0.50×10 ⁹	0.18×10 ⁹	0.30×10 ⁹	0.36×10 ⁹	0.49×10 ⁹
Skyline	25920	30201	44027	60859	37541	38148	46851	64488	44751	51471	56937	68753

Tabla E.8: Resultados por lotes para la configuración 2 (movimiento-N/A-20). Los algoritmos 2B2S y IB2S tienen valores similares porque no varian en este modo de ejecución.

Configuración: 2. Se utilizan 4 casos de prueba equivalentes por combinación.

Variante 5. Modo: actualización movimiento; %Cambio: no aplica; DistMax: 25

Algoritmo	Combinación											
	1	2	3	4	5	6	7	8	9	10	11	12
Tiempo promedio (ms)												
2B2S	21015	16771	69127	112789	37324	29349	66179	147464	60213	72817	115407	222870
IB2S	20980	16892	69342	112763	37348	29344	66204	147352	60296	72883	115666	223171
VC2S+	18126	25285	56352	88591	27009	30824	49997	101845	39281	44410	78862	122458
CD2S	10333	10959	25942	40349	20335	17199	26609	46532	26062	33310	54366	62202
Mejor tiempo (ms)												
2B2S	1228	2674	11979	23403	7483	6694	6920	24492	12834	14839	17602	37065
IB2S	1281	2587	12214	23469	7492	6659	6920	24379	12750	14603	17581	37192
VC2S+	2419	12725	24697	32245	5302	14675	22168	29437	6830	19747	19287	36710
CD2S	688	3907	7443	9455	3042	5888	6790	11938	4490	9451	9921	13071
Pior tiempo (ms)												
2B2S	148500	54509	359703	404758	153940	123939	378443	548117	249616	311049	414026	689441
IB2S	150037	54667	365181	393753	153720	123209	380433	543226	249159	311955	414856	695389
VC2S+	95505	48995	164222	191214	73661	74488	160467	339189	174471	149588	268721	317879
CD2S	55145	24777	83009	102878	100160	50643	110869	160686	127744	105222	201100	198182
vecesTMin (lotes)												
2B2S	0	2	0	0	0	0	0	0	0	0	0	0
IB2S	0	1	0	0	0	0	1	0	0	0	0	0
VC2S+	0	0	0	0	4	0	0	0	1	1	3	0
CD2S	40	37	40	40	36	40	39	40	39	39	37	40
vecesTMax (lotes)												
2B2S	5	1	3	10	15	5	12	17	16	15	16	19
IB2S	8	2	10	13	17	7	13	16	21	18	15	20
VC2S+	27	37	27	17	8	28	15	7	3	7	9	1
CD2S	0	0	0	0	0	0	0	0	0	0	0	0
#Comparaciones de dominancia												
2B2S	0.44×10 ⁹	0.50×10 ⁹	1.23×10 ⁹	1.95×10 ⁹	0.92×10 ⁹	0.90×10 ⁹	1.63×10 ⁹	2.84×10 ⁹	1.68×10 ⁹	2.02×10 ⁹	2.83×10 ⁹	4.57×10 ⁹
IB2S	0.44×10 ⁹	0.50×10 ⁹	1.23×10 ⁹	1.95×10 ⁹	0.92×10 ⁹	0.90×10 ⁹	1.63×10 ⁹	2.84×10 ⁹	1.68×10 ⁹	2.02×10 ⁹	2.83×10 ⁹	4.57×10 ⁹
VC2S+	0.14×10 ⁹	0.15×10 ⁹	0.38×10 ⁹	0.70×10 ⁹	0.19×10 ⁹	0.19×10 ⁹	0.38×10 ⁹	0.83×10 ⁹	0.25×10 ⁹	0.27×10 ⁹	0.53×10 ⁹	0.91×10 ⁹
CD2S	0.13×10 ⁹	0.15×10 ⁹	0.30×10 ⁹	0.45×10 ⁹	0.19×10 ⁹	0.21×10 ⁹	0.32×10 ⁹	0.58×10 ⁹	0.20×10 ⁹	0.30×10 ⁹	0.47×10 ⁹	0.66×10 ⁹
Skyline	28245	30027	46590	64262	39400	37680	48809	69130	46432	52292	62299	76275

Tabla E.9: Resultados por lotes para la configuración 2 (movimiento-N/A-25). Los algoritmos 2B2S y IB2S tienen valores similares porque no varian en este modo de ejecución.

Configuración: 2. Se utilizan 4 casos de prueba equivalentes por combinación.

Variante 1. Modo: actualización estatus; %Cambio: 10; DistMax: No aplica

Algoritmo	Combinación											
	1	2	3	4	5	6	7	8	9	10	11	12
Tiempo promedio (ms)												
2B2S	706	1171	3195	8413	1534	2475	4716	10441	3552	6470	10213	18177
IB2S	902	944	1679	3556	1145	1387	2299	4229	1896	2765	4270	6936
VC2S+	2229	2355	3369	4987	2955	2690	4202	5686	4231	4462	5988	7853
CD2S	95	1091	1701	2667	239	1380	2083	3105	545	2277	3124	4106
Mejor tiempo (ms)												
2B2S	46	419	1107	4832	515	1485	2075	4167	1231	3338	3837	8656
IB2S	594	610	773	2043	721	948	1148	1604	935	1541	1740	3292
VC2S+	1699	1896	2778	3524	2223	2358	3124	3013	2821	3415	3185	4451
CD2S	0	807	1059	1766	15	995	1331	1526	62	1501	1804	2372
Peor tiempo (ms)												
2B2S	1965	2312	4714	14558	2366	4089	7245	19089	5882	10160	17300	23402
IB2S	1292	1331	2492	5877	1519	2197	3343	7784	2865	4148	7066	8952
VC2S+	2919	2668	4210	7685	3541	3051	5981	8456	5726	6611	9025	10310
CD2S	310	1342	2254	3901	469	1777	2940	4746	977	3276	4728	5303
vecesTMin (lotes)												
2B2S	1	20	0	0	0	0	0	0	0	0	0	0
IB2S	0	12	20	1	0	22	20	4	0	2	8	0
VC2S+	0	0	0	0	0	0	0	0	0	0	0	0
CD2S	39	8	20	39	40	18	20	36	40	38	32	40
vecesTMax (lotes)												
2B2S	0	0	28	40	0	10	20	40	5	31	40	40
IB2S	0	0	0	0	0	0	0	0	0	0	0	0
VC2S+	40	40	12	0	40	30	20	0	35	9	0	0
CD2S	0	0	0	0	0	0	0	0	0	0	0	0
#Comparaciones de dominancia												
2B2S	0.44×10 ⁸	0.86×10 ⁸	2.29×10 ⁸	5.49×10 ⁸	1.10×10 ⁸	1.80×10 ⁸	3.51×10 ⁸	7.43×10 ⁸	2.74×10 ⁸	4.55×10 ⁸	7.33×10 ⁸	12.66×10 ⁸
IB2S	0.16×10 ⁸	0.29×10 ⁸	0.72×10 ⁸	1.58×10 ⁸	0.38×10 ⁸	0.62×10 ⁸	1.07×10 ⁸	2.14×10 ⁸	0.95×10 ⁸	1.54×10 ⁸	2.33×10 ⁸	3.87×10 ⁸
VC2S+	0.08×10 ⁸	0.19×10 ⁸	0.63×10 ⁸	1.44×10 ⁸	0.21×10 ⁸	0.34×10 ⁸	0.84×10 ⁸	1.82×10 ⁸	0.31×10 ⁸	0.71×10 ⁸	1.32×10 ⁸	2.15×10 ⁸
CD2S	0.03×10 ⁸	0.14×10 ⁸	0.44×10 ⁸	0.88×10 ⁸	0.09×10 ⁸	0.26×10 ⁸	0.54×10 ⁸	1.14×10 ⁸	0.12×10 ⁸	0.49×10 ⁸	0.85×10 ⁸	1.32×10 ⁸
Skyline	8861	12202	18522	28962	14408	16579	21286	29725	18499	23843	28887	36252

Tabla E.10: Resultados por lotes para la configuración 2 (cambioEstatus-10 %N/A)

Configuración: 2. Se utilizan 4 casos de prueba equivalentes por combinación.
 Variante 2. Modo: actualización estatus; %Cambio: 30; DistMax: no aplica

Algoritmo	Combinación											
	1	2	3	4	5	6	7	8	9	10	11	12
Tiempo promedio (ms)												
2B2S	636	1053	2975	7804	1378	2271	4280	9642	3225	5850	9314	16591
IB2S	1308	1447	2247	4279	1568	1800	2796	4916	2148	2868	4553	7267
VC2S+	2613	2777	3796	5221	3351	3110	4416	5872	4489	4595	6010	7753
CD2S	109	2117	2850	4280	252	2390	3309	4801	528	3129	4430	6310
Mejor tiempo (ms)												
2B2S	31	377	892	4791	451	1358	1910	4209	1136	2998	3463	7723
IB2S	1011	1152	1309	2807	1184	1459	1797	2309	1419	1981	2230	3648
VC2S+	2091	2468	3054	3917	2652	2793	3440	3522	3228	3619	3474	4711
CD2S	0	1824	2108	3195	78	2044	2579	2731	107	2542	2810	3806
Pior tiempo (ms)												
2B2S	1783	1996	4548	12543	2105	3619	6620	17005	5194	9003	15993	21229
IB2S	1638	1950	3716	6769	1968	2204	3666	7817	2997	3947	7097	9234
VC2S+	3323	3027	5791	7458	3850	3509	5541	8065	5792	6461	8801	10057
CD2S	330	2483	4300	6055	376	2743	3954	6735	900	4058	6503	7681
vecesTMin (lotes)												
2B2S	0	32	10	0	0	12	8	0	0	0	0	0
IB2S	0	8	30	22	0	28	32	21	0	33	16	6
VC2S+	0	0	0	0	0	0	0	0	0	0	0	0
CD2S	40	0	0	18	40	0	0	19	40	7	24	34
vecesTMax (lotes)												
2B2S	0	0	3	40	0	10	20	40	0	30	35	40
IB2S	0	0	0	0	0	0	0	0	0	0	0	0
VC2S+	40	40	37	0	40	30	20	0	40	10	5	0
CD2S	0	0	0	0	0	0	0	0	0	0	0	0
#Comparaciones de dominancia												
2B2S	0.41×10 ⁸	0.80×10 ⁸	2.15×10 ⁸	5.21×10 ⁸	1.02×10 ⁸	1.68×10 ⁸	3.32×10 ⁸	7.04×10 ⁸	2.53×10 ⁸	4.23×10 ⁸	6.88×10 ⁸	11.87×10 ⁸
IB2S	0.12×10 ⁸	0.26×10 ⁸	0.65×10 ⁸	1.57×10 ⁸	0.30×10 ⁸	0.54×10 ⁸	1.00×10 ⁸	2.14×10 ⁸	0.72×10 ⁸	1.27×10 ⁸	2.17×10 ⁸	3.71×10 ⁸
VC2S+	0.08×10 ⁸	0.18×10 ⁸	0.60×10 ⁸	1.38×10 ⁸	0.19×10 ⁸	0.32×10 ⁸	0.81×10 ⁸	1.73×10 ⁸	0.29×10 ⁸	0.66×10 ⁸	1.25×10 ⁸	2.03×10 ⁸
CD2S	0.03×10 ⁸	0.14×10 ⁸	0.45×10 ⁸	1.06×10 ⁸	0.08×10 ⁸	0.23×10 ⁸	0.57×10 ⁸	1.36×10 ⁸	0.11×10 ⁸	0.40×10 ⁸	0.90×10 ⁸	1.59×10 ⁸
Skyline	8639	11811	17987	28263	13861	16054	20749	29104	17848	23073	28067	35189

Tabla E.11: Resultados por lotes para la configuración 2 (cambioEstatus-30 %N/A)

Configuración: 2. Se utilizan 4 casos de prueba equivalentes por combinación.

Variante 3, Modo: actualización estatus; %Cambio: aleatorio; DistMax: no aplica

Algoritmo	Combinación											
	1	2	3	4	5	6	7	8	9	10	11	12
Tiempo promedio (ms)												
2B2S	675	1089	2942	7892	1402	2297	4202	9869	3254	5795	9293	16566
IB2S	1544	1764	2529	4994	1787	2116	3153	5659	2323	3223	5003	8090
VC2S+	2961	3074	3973	5541	3659	3371	4533	6195	4856	4829	6103	7909
CD2S	117	2620	3223	5254	264	2830	3794	5765	555	3598	4966	7476
Mejor tiempo (ms)												
2B2S	62	388	967	4851	376	1353	1763	4246	1107	3073	3434	8251
IB2S	1169	1273	1576	3261	1405	1560	2009	2712	1535	2217	2618	4538
VC2S+	2275	2743	3272	4181	2824	3065	3544	3693	3541	3668	3589	5008
CD2S	0	2180	2422	3808	62	2177	2669	3276	77	2676	3153	4774
Peor tiempo (ms)												
2B2S	2327	2184	4367	13177	2152	3920	7183	21185	5980	8923	16582	21314
IB2S	2044	2288	3523	7929	2158	2651	4196	10887	3678	4975	7771	10112
VC2S+	4819	3680	4622	7818	5118	3858	5806	10249	7429	6849	8738	10001
CD2S	341	3227	4074	7514	483	3489	4727	10031	1134	4650	7055	9327
vecesTMin (lotes)												
2B2S	0	36	10	0	0	20	18	0	0	0	0	0
IB2S	0	4	30	29	0	20	22	28	0	37	19	10
VC2S+	0	0	0	0	0	0	0	0	0	0	0	3
CD2S	40	0	0	11	40	0	0	12	40	3	21	27
vecesTMax (lotes)												
2B2S	0	0	0	0	40	0	10	20	40	0	30	40
IB2S	0	0	0	0	0	0	0	0	0	0	0	0
VC2S+	40	40	40	0	40	30	20	0	40	10	10	0
CD2S	0	0	0	0	0	0	0	0	0	0	0	0
#Comparaciones de dominancia												
2B2S	0.40×10 ⁸	0.80×10 ⁸	2.14×10 ⁸	5.18×10 ⁸	1.00×10 ⁸	1.67×10 ⁸	3.26×10 ⁸	6.95×10 ⁸	2.51×10 ⁸	4.17×10 ⁸	6.82×10 ⁸	11.75×10 ⁸
IB2S	0.12×10 ⁸	0.27×10 ⁸	0.70×10 ⁸	1.75×10 ⁸	0.30×10 ⁸	0.55×10 ⁸	1.08×10 ⁸	2.32×10 ⁸	0.73×10 ⁸	1.32×10 ⁸	2.35×10 ⁸	4.07×10 ⁸
VC2S+	0.08×10 ⁸	0.18×10 ⁸	0.60×10 ⁸	1.40×10 ⁸	0.19×10 ⁸	0.32×10 ⁸	0.80×10 ⁸	1.73×10 ⁸	0.29×10 ⁸	0.65×10 ⁸	1.24×10 ⁸	2.02×10 ⁸
CD2S	0.03×10 ⁸	0.15×10 ⁸	0.47×10 ⁸	1.22×10 ⁸	0.08×10 ⁸	0.23×10 ⁸	0.61×10 ⁸	1.49×10 ⁸	0.11×10 ⁸	0.41×10 ⁸	0.97×10 ⁸	1.80×10 ⁸
Skyline	8597	11783	17949	28223	13770	15961	20585	28867	17774	22895	27879	34984

Tabla E.12: Resultados por lotes para la configuración 2 (cambioEstatus-aleatorio-N/A)

Configuración: 2. Se utilizan 4 casos de prueba equivalentes por combinación.
 Variante 4. Modo: actualización estatus; %Cambio: 70; DistMax: no aplica

Algoritmo	Combinación											
	1	2	3	4	5	6	7	8	9	10	11	12
Tiempo promedio (ms)												
2B2S	632	1036	2859	7717	1371	2232	4058	9428	3129	5685	9049	16272
IB2S	1837	2259	3056	5802	2175	2718	3632	6490	2730	3923	5833	9284
VC2S+	3198	3463	4308	5895	3917	3824	4767	6512	5028	5194	6417	8266
CD2S	145	3534	4082	6408	286	3789	4519	6970	609	4559	6034	9008
Mejor tiempo (ms)												
2B2S	61	345	927	4850	420	1314	1811	4134	1081	2918	3339	8096
IB2S	1499	1884	2041	4019	1824	2306	2666	3480	1856	2963	3272	5319
VC2S+	2678	3182	3633	4697	3217	3465	4028	4242	3773	4243	4202	5457
CD2S	0	3151	3320	4992	47	3257	3836	4367	123	3748	4244	5899
Pior tiempo (ms)												
2B2S	1733	1997	4067	12404	2119	3513	6238	16133	4883	8609	15336	20482
IB2S	2200	2681	3963	8970	2480	3073	4618	9683	3512	5147	8568	12109
VC2S+	3980	3759	4824	8127	4367	4303	5519	8610	6288	6912	8707	10327
CD2S	423	3823	4870	9158	500	4211	5275	9532	1046	5414	8007	11620
vecesTMin (lotes)												
2B2S	0	40	15	0	0	30	20	0	0	7	8	0
IB2S	0	0	25	25	0	10	20	22	0	33	13	8
VC2S+	0	0	0	15	0	0	0	18	0	0	0	32
CD2S	40	0	0	0	40	0	0	0	40	0	19	0
vecesTMax (lotes)												
2B2S	0	0	0	0	20	0	20	30	0	30	30	40
IB2S	0	0	0	0	0	0	0	0	0	0	0	0
VC2S+	40	18	34	0	40	24	19	0	40	10	1	0
CD2S	0	22	6	20	0	16	1	10	0	0	9	0
#Comparaciones de dominancia												
2B2S	0.40×10 ⁸	0.79×10 ⁸	2.12×10 ⁸	5.15×10 ⁸	1.00×10 ⁸	1.66×10 ⁸	3.25×10 ⁸	6.91×10 ⁸	2.48×10 ⁸	4.15×10 ⁸	6.77×10 ⁸	11.71×10 ⁸
IB2S	0.13×10 ⁸	0.29×10 ⁸	0.76×10 ⁸	1.97×10 ⁸	0.34×10 ⁸	0.60×10 ⁸	1.18×10 ⁸	2.62×10 ⁸	0.81×10 ⁸	1.47×10 ⁸	2.63×10 ⁸	4.62×10 ⁸
VC2S+	0.08×10 ⁸	0.18×10 ⁸	0.60×10 ⁸	1.40×10 ⁸	0.19×10 ⁸	0.32×10 ⁸	0.80×10 ⁸	1.73×10 ⁸	0.28×10 ⁸	0.65×10 ⁸	1.24×10 ⁸	2.04×10 ⁸
CD2S	0.04×10 ⁸	0.16×10 ⁸	0.51×10 ⁸	1.38×10 ⁸	0.08×10 ⁸	0.25×10 ⁸	0.65×10 ⁸	1.69×10 ⁸	0.12×10 ⁸	0.44×10 ⁸	1.06×10 ⁸	2.07×10 ⁸
Skyline	8559	11717	17894	28164	13763	15899	20544	28818	17670	22834	27821	34998

Tabla E.13: Resultados por lotes para la configuración 2 (cambioEstatus-70 %N/A)

Configuración: 2. Se utilizan 4 casos de prueba equivalentes por combinación.
 Variante 5, Modo: actualización estatus; %Cambio: 90; DistMax: no aplica

Algoritmo	Combinación											
	1	2	3	4	5	6	7	8	9	10	11	12
Tiempo promedio (ms)												
2B2S	633	1046	2888	7778	1375	2249	4101	9486	3181	5761	9171	16406
IB2S	1946	2338	3284	6205	2201	2727	3750	6833	2775	4071	6224	9766
VC2S+	3296	3518	4430	6016	3976	3846	4831	6574	5146	5358	6640	8382
CD2S	155	3512	4268	6870	316	3754	4573	7350	630	4684	6430	9623
Mejor tiempo (ms)												
2B2S	46	337	995	4821	372	1338	1797	4145	1092	3007	3445	7927
IB2S	1604	1856	2234	4431	1812	2368	2621	3619	1930	2991	3308	5617
VC2S+	2683	3278	3869	4805	3232	3446	4148	4307	3934	4350	4213	5581
CD2S	16	3137	3496	5290	63	3383	3745	4557	157	3884	4180	6168
Peor tiempo (ms)												
2B2S	1729	1952	4134	12681	2123	3590	6268	16260	5104	8756	15633	20706
IB2S	2345	2887	4310	9441	2604	3054	4836	9985	3624	5120	9172	12031
VC2S+	4039	3870	4987	8206	4464	4320	5707	8518	6384	6943	9186	10343
CD2S	374	3905	5089	9832	495	4071	5447	10014	1048	5284	8692	11501
vecesTMin (lotes)												
2B2S	2	40	30	1	0	30	20	0	0	9	3	0
IB2S	0	0	10	20	0	10	20	19	0	31	17	1
VC2S+	0	0	0	19	0	0	0	21	0	0	2	39
CD2S	38	0	0	0	40	0	0	0	40	0	18	0
vecesTMax (lotes)												
2B2S	0	0	0	0	20	0	20	30	0	30	27	40
IB2S	0	0	0	0	0	0	0	0	0	0	0	0
VC2S+	40	23	27	0	40	28	20	0	40	10	5	0
CD2S	0	17	13	20	0	12	0	10	0	0	8	0
#Comparaciones de dominancia												
2B2S	0.40×10 ⁸	0.79×10 ⁸	2.12×10 ⁸	5.15×10 ⁸	1.01×10 ⁸	1.66×10 ⁸	3.26×10 ⁸	6.92×10 ⁸	2.49×10 ⁸	4.16×10 ⁸	6.78×10 ⁸	11.71×10 ⁸
IB2S	0.13×10 ⁸	0.30×10 ⁸	0.81×10 ⁸	2.14×10 ⁸	0.35×10 ⁸	0.63×10 ⁸	1.27×10 ⁸	2.83×10 ⁸	0.84×10 ⁸	1.54×10 ⁸	2.80×10 ⁸	4.98×10 ⁸
VC2S+	0.08×10 ⁸	0.18×10 ⁸	0.60×10 ⁸	1.42×10 ⁸	0.19×10 ⁸	0.32×10 ⁸	0.80×10 ⁸	1.75×10 ⁸	0.29×10 ⁸	0.65×10 ⁸	1.23×10 ⁸	2.05×10 ⁸
CD2S	0.04×10 ⁸	0.16×10 ⁸	0.54×10 ⁸	1.52×10 ⁸	0.09×10 ⁸	0.26×10 ⁸	0.71×10 ⁸	1.83×10 ⁸	0.13×10 ⁸	0.45×10 ⁸	1.14×10 ⁸	2.27×10 ⁸
Skyline	8531	11688	17842	28125	13735	15862	20520	28785	17636	22774	27730	34856

Tabla E.14: Resultados por lotes para la configuración 2 (cambioEstatus-90 %N/A)

Configuración: 3. Se utilizan 4 casos de prueba equivalentes por combinación.
Variante 3. Modo: combinado; %Cambio: aleatorio; DistMax: 10

Algoritmo	Combinación											
	1	2	3	4	5	6	7	8	9	10	11	12
Tiempo promedio (ms)												
2B2S	66482	229745	268246	235301	122979	259377	219443	527329	245659	250396	333451	631791
IB2S	48245	174689	207140	180922	89259	199283	164524	404092	181113	189290	256462	449637
VC2S+	31767	117193	128092	121194	55395	132774	94294	243605	85836	162301	160788	314727
CD2S	14199	48900	71119	76983	24201	66249	53271	148038	56780	66511	82838	188589
Mejor tiempo (ms)												
2B2S	8519	17835	11982	27866	8218	55026	25382	277007	65082	71509	107151	376346
IB2S	7097	14542	9080	23304	6761	40833	18.94	227778	41949	53004	77645	313547
VC2S+	3951	12704	11341	21620	5534	28435	13212	102305	18678	30775	44876	147918
CD2S	1073	5897	4693	14490	1326	14698	6040	58999	9420	12003	24075	107127
Pior tiempo (ms)												
2B2S	164107	564266	711691	512681	333263	602426	373581	855389	650819	473431	743232	967966
IB2S	117652	482637	592743	392344	232657	515967	298280	713129	502496	396334	609368	701715
VC2S+	82921	273154	369750	427257	178201	325116	232817	549686	236925	362668	413513	575160
CD2S	39867	107715	202455	280813	72914	192242	114523	317529	161239	173100	168223	297429
vecesTMin (lotes)												
2B2S	0	0	0	0	0	0	0	0	0	0	0	0
IB2S	0	0	0	0	0	0	0	0	0	0	0	0
VC2S+	0	0	0	0	0	0	0	0	0	0	0	0
CD2S	40	40	40	40	40	40	40	40	40	40	40	40
vecesTMax (lotes)												
2B2S	40	40	40	40	40	40	40	40	40	40	40	40
IB2S	0	0	0	0	0	0	0	0	0	0	0	0
VC2S+	0	0	0	0	0	0	0	0	0	0	0	0
CD2S	0	0	0	0	0	0	0	0	0	0	0	0
#Comparaciones de dominancia												
2B2S	0.88×10 ⁹	1.84×10 ⁹	2.25×10 ⁹	2.40×10 ⁹	1.55×10 ⁹	2.67×10 ⁹	4.73×10 ⁹	3.49×10 ⁹	3.75×10 ⁹	4.48×10 ⁹	6.98×10 ⁹	
IB2S	0.66×10 ⁹	1.45×10 ⁹	1.76×10 ⁹	1.90×10 ⁹	1.17×10 ⁹	2.08×10 ⁹	2.07×10 ⁹	3.76×10 ⁹	2.64×10 ⁹	3.54×10 ⁹	5.18×10 ⁹	
VC2S+	0.17×10 ⁹	0.46×10 ⁹	0.59×10 ⁹	0.84×10 ⁹	0.21×10 ⁹	0.64×10 ⁹	0.61×10 ⁹	1.37×10 ⁹	0.25×10 ⁹	0.73×10 ⁹	0.87×10 ⁹	1.62×10 ⁹
CD2S	0.08×10 ⁹	0.15×10 ⁹	0.25×10 ⁹	0.42×10 ⁹	0.09×10 ⁹	0.21×10 ⁹	0.26×10 ⁹	0.66×10 ⁹	0.13×10 ⁹	0.21×10 ⁹	0.34×10 ⁹	0.69×10 ⁹
Skyline	51954	80953	85315	88327	60466	87677	83445	117586	86163	89439	100721	133404

Tabla E.15: Resultados por lotes para la configuración 3 (combinado-aleatorio-10)

Configuración: 3. Se utilizan 4 casos de prueba equivalentes por combinación.
 Variante 4. Modo: combinado; %Cambio: aleatorio; DistMax: 20

Algoritmo	Combinación											
	1	2	3	4	5	6	7	8	9	10	11	12
Tiempo promedio (ms)												
2B2S	86667	246172	289844	278459	133596	292922	232155	538819	250533	261016	367633	676182
IB2S	62804	182068	219054	219184	99201	218773	173425	406708	186835	193164	281194	527595
VC2S+	43490	135046	147205	155363	57923	172959	100546	265887	110371	165960	211742	334121
CD2S	20601	61212	86596	96214	28340	85497	55269	165797	62314	71494	102723	197758
Mejor tiempo (ms)												
2B2S	9270	22862	9848	37664	10563	54100	31059	206702	57726	83713	113081	422008
IB2S	7475	17852	7733	30344	8704	38542	24324	168853	43737	53599	81259	325010
VC2S+	5242	15734	8906	29063	7752	26261	22390	92106	25070	50089	56864	120758
CD2S	1558	7634	3993	19512	2293	14256	10138	65254	12072	19961	25060	64880
Pior tiempo (ms)												
2B2S	360300	646959	745611	672920	399220	684533	506268	982803	512826	534659	773917	1013458
IB2S	249864	489112	561546	603508	293454	578551	420950	770155	400154	395261	625408	838830
VC2S+	195699	377254	387679	419184	200010	476567	264548	611212	230163	307738	525968	569383
CD2S	85315	143225	221226	307967	83302	207430	132970	375196	136125	147312	244251	348956
vecesTMin (lotes)												
2B2S	0	0	0	0	0	0	0	0	0	0	0	0
IB2S	0	0	0	0	0	0	0	0	0	0	0	0
VC2S+	0	0	0	0	0	0	0	0	0	0	0	0
CD2S	40	40	40	40	40	40	40	40	40	40	40	40
vecesTMax (lotes)												
2B2S	40	39	39	40	40	40	40	40	40	39	40	40
IB2S	0	0	0	0	0	0	0	0	0	0	0	0
VC2S+	0	1	1	0	0	0	0	0	1	0	0	0
CD2S	0	0	0	0	0	0	0	0	0	0	0	0
#Comparaciones de dominancia												
2B2S	1.03×10 ⁹	1.86×10 ⁹	2.29×10 ⁹	2.62×10 ⁹	1.64×10 ⁹	2.83×10 ⁹	2.76×10 ⁹	4.81×10 ⁹	3.43×10 ⁹	3.87×10 ⁹	4.72×10 ⁹	7.03×10 ⁹
IB2S	0.77×10 ⁹	1.45×10 ⁹	1.77×10 ⁹	2.09×10 ⁹	1.26×10 ⁹	2.17×10 ⁹	2.15×10 ⁹	3.76×10 ⁹	2.63×10 ⁹	3.71×10 ⁹	5.63×10 ⁹	
VC2S+	0.22×10 ⁹	0.54×10 ⁹	0.65×10 ⁹	0.99×10 ⁹	0.22×10 ⁹	0.67×10 ⁹	0.68×10 ⁹	1.45×10 ⁹	0.27×10 ⁹	0.83×10 ⁹	0.96×10 ⁹	1.68×10 ⁹
CD2S	0.10×10 ⁹	0.17×10 ⁹	0.28×10 ⁹	0.50×10 ⁹	0.10×10 ⁹	0.24×10 ⁹	0.28×10 ⁹	0.70×10 ⁹	0.14×10 ⁹	0.24×10 ⁹	0.38×10 ⁹	0.71×10 ⁹
Skyline	57091	83794	87388	94050	62202	91899	85117	119213	86394	92466	104428	134111

Tabla E.16: Resultados por lotes para la configuración 3 (combinado-aleatorio-20)

Configuración: 4. Se utilizan 2 casos de prueba equivalentes por combinación.

Variante 1. Modo: combinado; %Cambio: aleatorio; DistMax: 10

Algoritmo					Combinación							
	1	2	3	4	5	6	7	8	9	10	11	12
Tiempo promedio (ms)												
2B2S	19053	8433	57842	208442	41502	76444	136000	321767	63739	87582	173093	283167
IB2S	25588	18440	43884	142184	34886	55041	95558	212251	64659	62663	129836	195279
VC2S+	38894	43252	92637	173614	44243	102574	159564	238913	75141	92440	121673	161863
CD2S	4812	34731	58344	118604	8151	51839	84432	158571	23546	57904	101016	128372
Mejor tiempo (ms)												
2B2S	4074	2714	15491	102633	28171	37377	62327	127469	8393	49281	47619	147111
IB2S	17364	11981	22475	75634	26054	37144	55791	91647	25605	44116	39859	104266
VC2S+	22510	28626	35140	80823	30107	58379	82914	123846	39530	45860	57458	79311
CD2S	1057	24553	27758	58053	4491	32358	51634	83276	4740	34356	40875	65365
Pior tiempo (ms)												
2B2S	77155	21160	133742	369659	64538	122581	280454	929892	127688	170911	301578	510959
IB2S	41933	24973	83526	245057	45131	88202	156734	572146	117123	92915	230726	300086
VC2S+	87657	65848	146021	310315	56394	181445	243420	610090	122168	154113	192440	305339
CD2S	16164	45755	86379	200556	13138	70820	127514	404626	44057	85363	173288	241586
vecesTMin (lotes)												
2B2S	0	20	3	0	0	1	0	0	0	3	0	0
IB2S	0	0	17	4	0	8	2	1	0	4	10	0
VC2S+	0	0	0	0	0	0	0	0	0	0	2	4
CD2S	20	0	0	0	16	20	11	18	19	20	13	8
vecesTMax (lotes)												
2B2S	0	0	1	18	7	9	2	20	10	5	12	20
IB2S	0	0	0	0	0	0	0	0	0	1	0	0
VC2S+	20	16	19	2	13	11	18	0	10	13	8	0
CD2S	0	4	0	0	0	0	0	0	0	1	0	0
#Comparaciones de dominancia												
2B2S	0.41×10 ⁹	0.35×10 ⁹	1.23×10 ⁹	3.23×10 ⁹	0.97×10 ⁹	1.74×10 ⁹	2.78×10 ⁹	4.90×10 ⁹	1.81×10 ⁹	2.39×10 ⁹	3.89×10 ⁹	6.08×10 ⁹
IB2S	0.117×10 ⁹	0.117×10 ⁹	0.61×10 ⁹	1.66×10 ⁹	0.40×10 ⁹	0.87×10 ⁹	1.45×10 ⁹	2.64×10 ⁹	0.86×10 ⁹	1.14×10 ⁹	2.07×10 ⁹	3.38×10 ⁹
VC2S+	0.09×10 ⁹	0.10×10 ⁹	0.42×10 ⁹	1.17×10 ⁹	0.10×10 ⁹	0.49×10 ⁹	1.35×10 ⁹	1.45×10 ⁹	0.11×10 ⁹	0.52×10 ⁹	0.61×10 ⁹	1.24×10 ⁹
CD2S	0.03×10 ⁹	0.06×10 ⁹	0.22×10 ⁹	0.78×10 ⁹	0.04×10 ⁹	0.17×10 ⁹	0.50×10 ⁹	0.87×10 ⁹	0.07×10 ⁹	0.16×10 ⁹	0.39×10 ⁹	0.87×10 ⁹
Skyline	37960	30704	57659	84377	51484	70139	82819	101070	62935	69462	84533	100919

Tabla E.17: Resultados por lotes para la configuración 4 (combinado-aleatorio-10)

Configuración: 5. Se utilizan 2 casos de prueba equivalentes por combinación.

Variante 2. Modo: combinado; %Cambio: aleatorio; DistMax: 10

Algoritmo	Combinación											
	1	2	3	4	5	6	7	8	9	10	11	12
Tiempo promedio (ms)												
2B2S	1072048	451006	235305	925080	1379705	1034690	1903999	1290739	991232	721955	5673676	2860503
IB2S	567881	288025	141932	755086	894438	572215	1214875	897180	495515	392776	4020440	2190155
VC2S+	787254	556885	227046	823104	788926	588183	1528271	851537	601975	533264	4950607	2298177
CD2S	305533	195295	110849	522772	487162	334948	729740	518568	276803	250531	2463394	1335759
Mejor tiempo (ms)												
2B2S	581693	68316	61198	461947	80339	579499	916905	713651	491600	436036	3649845	1606228
IB2S	367678	61526	66612	380751	73460	383725	722169	481028	241792	196966	2821858	1272180
VC2S+	303532	143797	70091	451417	78377	369486	717587	417223	286791	265449	2436226	1330580
CD2S	134754	75708	55335	354715	24538	179635	355400	306123	131427	172521	1226972	73207
Pior tiempo (ms)												
2B2S	1813877	815444	598789	1992045	3833599	1863889	4745741	2386615	1562127	1194793	8315335	4581509
IB2S	1033831	587687	350685	1775065	2096834	963926	2576367	1858778	726340	635956	5711089	3562766
VC2S+	1863858	1082272	530228	1436980	2223753	1010811	3837220	1324031	817020	1024346	8027875	3948443
CD2S	669551	321904	216269	770050	1342505	592848	1553340	747022	416694	365758	4118904	2506612
vecesTMin (lotes)												
2B2S	0	0	1	0	0	0	0	0	0	0	0	0
IB2S	0	3	1	2	0	0	0	0	0	0	0	0
VC2S+	0	0	0	0	0	0	0	0	0	0	0	0
CD2S	20	17	18	18	20	20	20	20	20	20	20	20
vecesTMax (lotes)												
2B2S	10	2	13	13	14	20	16	20	20	17	14	19
IB2S	0	0	0	0	0	0	0	0	0	0	0	0
VC2S+	10	18	7	7	6	0	4	0	0	3	6	1
CD2S	0	0	0	0	0	0	0	0	0	0	0	0
#Comparaciones de dominancia												
2B2S	0.54×10 ¹⁰	0.32×10 ¹⁰	0.24×10 ¹⁰	0.57×10 ¹⁰	0.69×10 ¹⁰	0.74×10 ¹⁰	1.07×10 ¹⁰	0.94×10 ¹⁰	0.95×10 ¹⁰	0.80×10 ¹⁰	2.92×10 ¹⁰	1.98×10 ¹⁰
IB2S	0.23×10 ¹⁰	0.17×10 ¹⁰	0.13×10 ¹⁰	0.41×10 ¹⁰	0.33×10 ¹⁰	0.36×10 ¹⁰	0.60×10 ¹⁰	0.57×10 ¹⁰	0.40×10 ¹⁰	0.39×10 ¹⁰	1.68×10 ¹⁰	1.28×10 ¹⁰
VC2S+	0.12×10 ¹⁰	0.14×10 ¹⁰	0.11×10 ¹⁰	0.31×10 ¹⁰	0.07×10 ¹⁰	0.11×10 ¹⁰	0.27×10 ¹⁰	0.27×10 ¹⁰	0.08×10 ¹⁰	0.13×10 ¹⁰	0.50×10 ¹⁰	0.49×10 ¹⁰
CD2S	0.04×10 ¹⁰	0.04×10 ¹⁰	0.04×10 ¹⁰	0.16×10 ¹⁰	0.03×10 ¹⁰	0.05×10 ¹⁰	0.09×10 ¹⁰	0.12×10 ¹⁰	0.03×10 ¹⁰	0.05×10 ¹⁰	0.17×10 ¹⁰	0.22×10 ¹⁰
Skyline	147215	115995	93224	155385	150004	152483	188406	164996	145736	128343	284818	226104

Tabla E.18: Resultados por lotes para la configuración 5 (combinado-aleatorio-10)

Configuración: 6. Se utilizan 2 casos de prueba equivalentes por combinación.

Variante 3. Modo: combinado; %Cambio: aleatorio; DistMax: 10

Algoritmo	Combinación											
	1	2	3	4	5	6	7	8	9	10	11	12
Tiempo promedio (ms)												
2B2S	623501	594818	1434132	3513093	6233029	5114734	4190726	21298121	22296622	15680003	12048595	17576143
IB2S	317669	300634	823946	2388917	3617274	3301025	2655691	15368520	14729550	11237405	8933723	13230199
VC2S+	443539	311035	789166	1988047	3756815	4522303	2956791	13238388	16728690	15701928	8894275	11657987
CD2S	178498	224536	512137	1320372	2363211	1961163	1802195	92557560	11700756	5879871	4720653	8521347
Mejor tiempo (ms)												
2B2S	277132	80820	345258	1959722	4062311	2649010	2177233	3964559	8251307	10136108	5164888	7615572
IB2S	93588	37101	252580	1146041	2407253	1686738	1220653	2068092	5401001	6374918	3501975	5431792
VC2S+	106112	40450	224045	700924	2318323	2021541	1086421	1329322	4231849	5868381	2292954	4304668
CD2S	46957	32284	130591	548045	1144978	969681	777976	943270	3324367	2466007	1673693	2589529
Pior tiempo (ms)												
2B2S	15558596	1606037	3128797	5622860	1093895	10250934	5763856	44048071	41069524	23402878	21949348	29958387
IB2S	830031	834124	1817268	3705427	6764612	7594047	3556908	33663735	30131904	17931923	18228585	22160681
VC2S+	1095421	876477	1562010	3582420	5942173	7813540	5168415	33023979	46746686	30551609	15366853	22193209
CD2S	477202	619377	1098384	2392796	4844586	3831460	2540464	21768259	29464284	10412411	9874916	29852316
vecesTMin (lotes)												
2B2S	0	0	0	0	0	0	0	0	0	0	0	0
IB2S	0	2	0	0	0	0	0	0	2	0	0	0
VC2S+	0	4	0	0	0	0	0	0	1	0	0	1
CD2S	20	14	20	20	20	20	20	20	17	20	20	19
vecesTMax (lotes)												
2B2S	17	20	15	20	20	16	20	20	18	13	20	20
IB2S	0	0	0	0	0	0	0	0	0	0	0	0
VC2S+	3	0	5	0	0	4	0	0	2	7	0	0
CD2S	0	0	0	0	0	0	0	0	0	0	0	0
#Comparaciones de dominancia												
2B2S	0.40×10 ¹⁰	0.35×10 ¹⁰	0.75×10 ¹⁰	1.46×10 ¹⁰	2.24×10 ¹⁰	2.01×10 ¹⁰	1.93×10 ¹⁰	4.93×10 ¹⁰	6.71×10 ¹⁰	5.61×10 ¹⁰	4.66×10 ¹⁰	6.53×10 ¹⁰
IB2S	0.16×10 ¹⁰	0.16×10 ¹⁰	0.38×10 ¹⁰	0.85×10 ¹⁰	1.03×10 ¹⁰	1.06×10 ¹⁰	1.05×10 ¹⁰	3.17×10 ¹⁰	3.53×10 ¹⁰	3.02×10 ¹⁰	2.76×10 ¹⁰	3.97×10 ¹⁰
VC2S+	0.07×10 ¹⁰	0.09×10 ¹⁰	0.27×10 ¹⁰	0.50×10 ¹⁰	0.41×10 ¹⁰	0.48×10 ¹⁰	0.41×10 ¹⁰	0.58×10 ¹⁰	1.14×10 ¹⁰	0.62×10 ¹⁰	1.03×10 ¹⁰	0.40×10 ¹⁰
CD2S	0.03×10 ¹⁰	0.05×10 ¹⁰	0.11×10 ¹⁰	0.22×10 ¹⁰	0.09×10 ¹⁰	0.15×10 ¹⁰	0.21×10 ¹⁰	0.43×10 ¹⁰	0.24×10 ¹⁰	0.19×10 ¹⁰	0.19×10 ¹⁰	0
Skyline	123084	101678	157716	231891	286004	267407	244309	397250	450562	401697	356645	415480

Tabla E.19: Resultados por lotes para la configuración 6 (combinado-aleatorio-10)

Configuración: 7. Se utilizan 4 casos de prueba equivalentes por combinación.

Variante 1. Modo: combinado; %Cambio: aleatorio; DistMax: 50

Algoritmo	Combinación (Conf 7a)					Combinación (Conf7b)				
	13	14	15	16	17	18	19	20		
Tiempo promedio (ms)										
2B2S	-	-	-	-	-	-	-	-	-	-
IB2S	1083033	1765696	1785413	2624414	1624037	2339024	3307094	3241184		
VC2S+	1674572	2545413	2508128	3667202	2014814	2881082	3720199	372035		
CD2S	762177	1137067	1191072	1642219	1153973	1634939	2148996	2139835		
Mejor tiempo (ms)										
2B2S	-	-	-	-	-	-	-	-	-	-
IB2S	329659	333823	422972	653515	168332	28873	877412	1111879		
VC2S+	397700	469610	680056	750285	263062	497358	917567	1094291		
CD2S	161612	162435	217389	29284	190836	251849	613698	680443		
Peor tiempo (ms)										
2B2S	-	-	-	-	-	-	-	-	-	-
IB2S	4669793	4606955	5338388	5473659	6365037	9999451	9372119	11746583		
VC2S+	5499555	8150061	7153243	8084661	7609900	11384955	11280215	14097032		
CD2S	3449010	3120296	3510697	3760438	4059183	6316653	5409409	7249892		
vecesTMin (lotes)										
2B2S	-	-	-	-	-	-	-	-	-	-
IB2S	0	0	0	0	10	0	0	0	0	0
VC2S+	0	0	0	0	0	0	0	0	0	0
CD2S	40	40	40	40	30	40	40	40	40	40
vecesTMax (lotes)										
2B2S	-	-	-	-	-	-	-	-	-	-
IB2S	0	2	1	2	4	5	14	8		
VC2S+	40	38	39	38	36	35	26	32		
CD2S	0	0	0	0	0	0	0	0	0	0
#Comparaciones de dominancia										
2B2S	-	-	-	-	-	-	-	-	-	-
IB2S	0.75×10^{10}	1.16×10^{10}	1.37×10^{10}	1.91×10^{10}	1.00×10^{10}	1.47×10^{10}	2.13×10^{10}	2.23×10^{10}		
VC2S+	0.13×10^{10}	0.18×10^{10}	0.17×10^{10}	0.20×10^{10}	0.28×10^{10}	0.34×10^{10}	0.40×10^{10}	0.37×10^{10}		
CD2S	0.08×10^{10}	0.11×10^{10}	0.11×10^{10}	0.12×10^{10}	0.13×10^{10}	0.16×10^{10}	0.18×10^{10}	0.18×10^{10}		
Skyline	177879	208045	209641	238735	190920	219462	248962	244130		

Tabla E.20: Resultados por lotes para la configuración 7 (combinado-aleatorio-50)

APÉNDICE F

SIMULACIONES

Exp	Conf	%Cambio (objetos)	DistMax (m)	Modo ejecución	Ejecuciones (cas×lot×sim×alg)	Total
I	1	aleatorio	10	combinado	$48 \times 10 \times 50 \times 4$	96.000
	1	aleatorio	20	combinado	$48 \times 10 \times 50 \times 4$	96.000
II	2	aleatorio	10	combinado	$48 \times 10 \times 50 \times 4$	96.000
	2	aleatorio	20	combinado	$48 \times 10 \times 50 \times 4$	96.000
	3	aleatorio	10	combinado	$48 \times 10 \times 50 \times 4$	96.000
	3	aleatorio	20	combinado	$48 \times 10 \times 50 \times 4$	96.000
III	2		5	actualización movimiento	$48 \times 10 \times 50 \times 4$	96.000
	2		10	actualización movimiento	$48 \times 10 \times 50 \times 4$	96.000
	2		15	actualización movimiento	$48 \times 10 \times 50 \times 4$	96.000
	2		20	actualización movimiento	$48 \times 10 \times 50 \times 4$	96.000
	2		25	actualización movimiento	$48 \times 10 \times 50 \times 4$	96.000
IV	2	10 %		actualización estatus	$48 \times 10 \times 50 \times 4$	96.000
	2	30 %		actualización estatus	$48 \times 10 \times 50 \times 4$	96.000
	2	aleatorio		actualización estatus	$48 \times 10 \times 50 \times 4$	96.000
	2	70 %		actualización estatus	$48 \times 10 \times 50 \times 4$	96.000
	2	90 %		actualización estatus	$48 \times 10 \times 50 \times 4$	96.000
V	4	aleatorio	10	combinado	$24 \times 10 \times 50 \times 4$	48.000
	5	aleatorio	10	combinado	$24 \times 10 \times 50 \times 4$	48.000
	6	aleatorio	10	combinado	$24 \times 10 \times 50 \times 4$	48.000
VI	7	aleatorio	50	combinado	$32 \times 10 \times 50 \times 3$	48.000
Total de ejecuciones						1.728.000

Tabla F.1: Ejecuciones realizadas

El detalle de las simulaciones pueden observarse en la Tabla F.1. Donde:

- * **Exp:** indica el experimento para el cual se realizó el estudio.
- * **Conf:** corresponde a la configuración ejecutada.
- * **%Cambio:** es el parámetro utilizado para establecer el porcentaje de localidades que

cambian de estatus en el plano en cada simulación, los objetos que cambian de estatus son escogidos por el simulador de forma aleatoria hasta cubrir el valor dado

- * **DistMax:** limita la distancia máxima que el punto de referencia móvil se puede desplazar en una simulación.
- * **Modo ejecución:** establece el modo en que son ejecutados los algoritmos según el simulador, puede ser: *modo actualización de estatus* en todas las simulaciones, *modo actualización de desplazamiento* en todas las simulaciones, o combinado, donde aleatoriamente el simulador escoge en cada simulación el cambio a realizar. El *modo inicial* siempre se ejecutará en la primera simulación para inicializar los algoritmos con el caso de prueba.
- * **Ejecuciones:** indica cuantas pruebas se realizaron por configuración para estudiar los algoritmos: **cas** se refiere a la cantidad de casos de prueba ejecutados, **lot** la cantidad de lotes de simulaciones realizadas, **sim** las simulaciones efectuadas y **alg** el número de algoritmos utilizados en la simulación.

Como para cada combinación `#ptoRef - #dimNoEsp` se crearon varios casos de prueba distintos, entonces cada combinación es probada 2.000 veces para las configuraciones 1,2 y 3 con cada algoritmo y configuración, al menos 1.000 veces para las configuraciones 4, 5 y 6 con cada algoritmo y configuración y 2.000 veces para la configuración 7 excluyendo el algoritmo 2B2S. Este calculo es el resultado de multiplicar: 4 casos de prueba \times 10 lotes \times 50 simulaciones y 2 casos \times 10 lotes \times 50 simulaciones respectivamente.

Se realizan una menor cantidad de pruebas para las configuraciones 4, 5 y 6 (Experimento V) debido a que son mas costosas en tiempo de ejecución por la cantidad de objetos de entrada. Aunque se probó una cantidad mayor de veces las combinaciones para la configuración 7, en total se hicieron la misma cantidad de simulaciones porque se crearon menos combinaciones por ser un conjunto de datos aún mayor.

APÉNDICE G

PROPIEDADES ESPACIALES

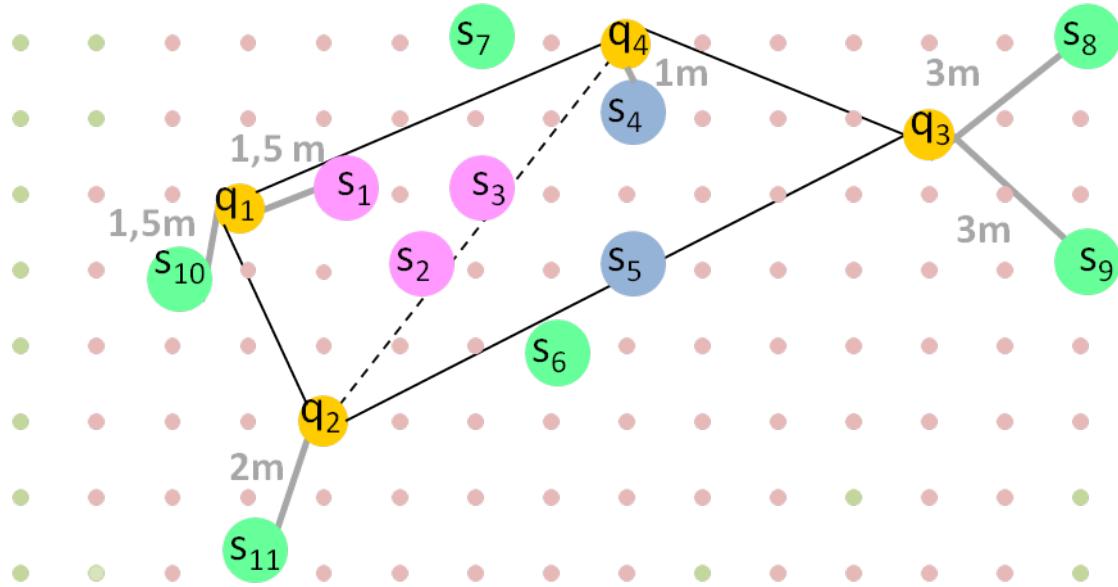


Figura G.1: Propiedades espaciales

En la Figura G.1, se muestra el *skyline* espacial en base a los objetos disponibles en el plano, donde:

- ★ Los puntos rojos representan los puestos ocupados y los verdes los disponibles que no forman parte del *skyline*.
- ★ Los círculos medianos amarillos son los puntos de referencia: q_1, q_2, q_3, q_4 . El punto q_3 es el punto de referencia móvil.
- ★ Las líneas sólidas negras forman el polígono del *convex hull* para los algoritmos VC2S+ y CD2S.
- ★ Sólo para el algoritmo CD2S, la linea punteada negra divide el *convex hull* en dos áreas: el triángulo de la izquierda (q_1, q_2, q_4) es el *convex hull* estático y el triángulo

de la derecha (q_2 , q_3 , q_4) es la región del *convex hull* que se vería afectada por el movimiento de q_3 .

* Los círculos verdes, rosados y azules representan al *skyline* espacial:

- ▷ Para el algoritmo **CD2S** forman: el *skyline* espacial fuera del *convex hull* (s_6 , s_7 , s_8 , s_9 , s_{10} y s_{11}), el *skyline* estático (s_1 , s_2 y s_3) y el *skyline* dinámico (s_4 y s_5), respectivamente. Cabe destacar que a los objetos dentro del *convex hull* no se les realiza la verificación de dominancia para determinar su pertenencia al *skyline*, una vez encontrados como disponibles son agregados automáticamente al *skyline*. En caso de que el punto de referencia q_3 se haya movido y por ende la forma del *convex hull* haya cambiado, los objetos s_1 , s_2 y s_3 continúan en el *skyline* y no se les realiza verificación de dominancia.
- ▷ Para el algoritmo **VC2S+** los círculos rosados serían reemplazados por azules generando sólo dos grupos: el *skyline* fuera del *convex hull* (s_1 , s_2 y s_3) y el *skyline* dentro del *convex hull* (s_1 , s_2 , s_3 , s_4 y s_5). Al igual que le algoritmo **CD2S**, los objetos dentro del *convex hull* no se les realiza la verificación de dominancia.
- ▷ Para los algoritmos **2B2S** y **IB2S** los círculos rosados y azules serían reemplazados por verdes generando un sólo grupo: el *skyline* espacial (s_1 , s_2 , s_3 , s_4 , s_5 , s_6 , s_7 , s_8 , s_9 , s_{10} y s_{11}).

* Las líneas grises denotan las distancias entre un punto de referencia y el objeto disponible más cercano.

Se observa que dos objetos disponibles, s_1 y s_{10} , pueden estar a la misma distancia del punto de referencia. Estos objetos, además tienen el mínimo valor entre los demás disponibles en la dimensión del punto de referencia espacial 1,5m y como ninguno domina a otro en las demás dimensiones, porque s_1 está más cerca que s_{10} de q_4 , pero s_{10} está más cerca que s_1 de q_2 , entonces son incomparables entre sí y ambos pertenecen al *skyline*. Este caso también se da con los objetos s_8 y s_9 que están a la misma distancia de q_3 , 3m la cual es la distancia mínima para esa dimensión, ya que los objetos que están más cerca están ocupados y son incomparables entre sí porque s_8 está más cerca de q_4 y s_9 de q_2 .

Para los puntos de referencia q_4 y q_2 , sólo existe un objeto con la distancia mínima, s_4 y s_{11} respectivamente, por lo tanto ambos forman parte del *skyline*.

Los objetos s_6 y s_7 , aunque no tienen un mínimo en alguna dimensión, tienen una buena combinación de valores que los hace formar parte del *skyline*. El objeto s_6 es dominado parcialmente en la dimensión correspondiente al punto de referencia q_3 por s_5 , s_4 , s_8 y s_9 , pero a la vez, esos objetos son dominados parcialmente por s_6 en la dimensión referente al punto q_3 , análogamente ocurre con s_2 y s_{11} . De la misma manera, el objeto s_7 no es dominado por s_4 porque aunque s_4 es el mejor con respecto a q_4 , está más lejos de q_1 que s_7 , análogamente ocurre con s_1 ; podría estar dominado por s_3 o s_2 , pero ambos están más lejos de q_4 que s_7 .

Los objetos s_1 , s_2 , s_3 , s_4 y s_5 , sólo por estar en el *convex hull*, forman parte del *skyline*, ya que ninguno es mejor que otro considerando todas las dimensiones.

Otra característica que se observa, es que si dos puntos de referencia están cerca, por ejemplo q_1 y q_2 , los objetos disponibles más cercanos a esos puntos s_1 y s_{10} para q_1 , y s_{11} para q_2 , también están relativamente cerca. Esta característica deja de aparecer cuando hay muchos objetos ocupados cercanos a estos puntos, como ocurre con los puntos de referencia q_3 y q_4 y los objetos s_4 y s_8 y s_9 .

APÉNDICE H

RESULTADOS COMPLEMENTARIOS

H.1 Experimento I

H.1.1 Tiempo detallado

En la Figura H.1 se muestran las gráficas por modo de ejecución de ambas variantes, donde se desglosa el tiempo promedio de los algoritmos para las distintas combinaciones, el eje x indica las combinaciones y el y el tiempo promedio. Considerando los tiempos de ejecución para sólo el *modo actualización de desplazamiento* (t_{Mov}), se puede observar que el algoritmo CD2S tiene el menor tiempo de ejecución, sin embargo, en segundo lugar está el algoritmo VC2S+ y no el IB2S como lo es totalizando los tiempos de los modos de ejecución, ya que el tiempo de ejecución en el *modo actualización de estatus* (t_{Est}) para VC2S+ es mas alto que para el resto de los algoritmos. El motivo de este comportamiento es que el algoritmo VC2S+ fue diseñado para realizar la actualización del conjunto *skyline* sólo para el *modo actualización de desplazamiento* y no en combinación con el *modo actualización de estatus*.

En general, los algoritmos se demoran un tiempo similar en la simulación inicial t_{Ini} , que consiste en preparar las estructuras y devolver el *skyline* correspondiente a las condiciones iniciales.

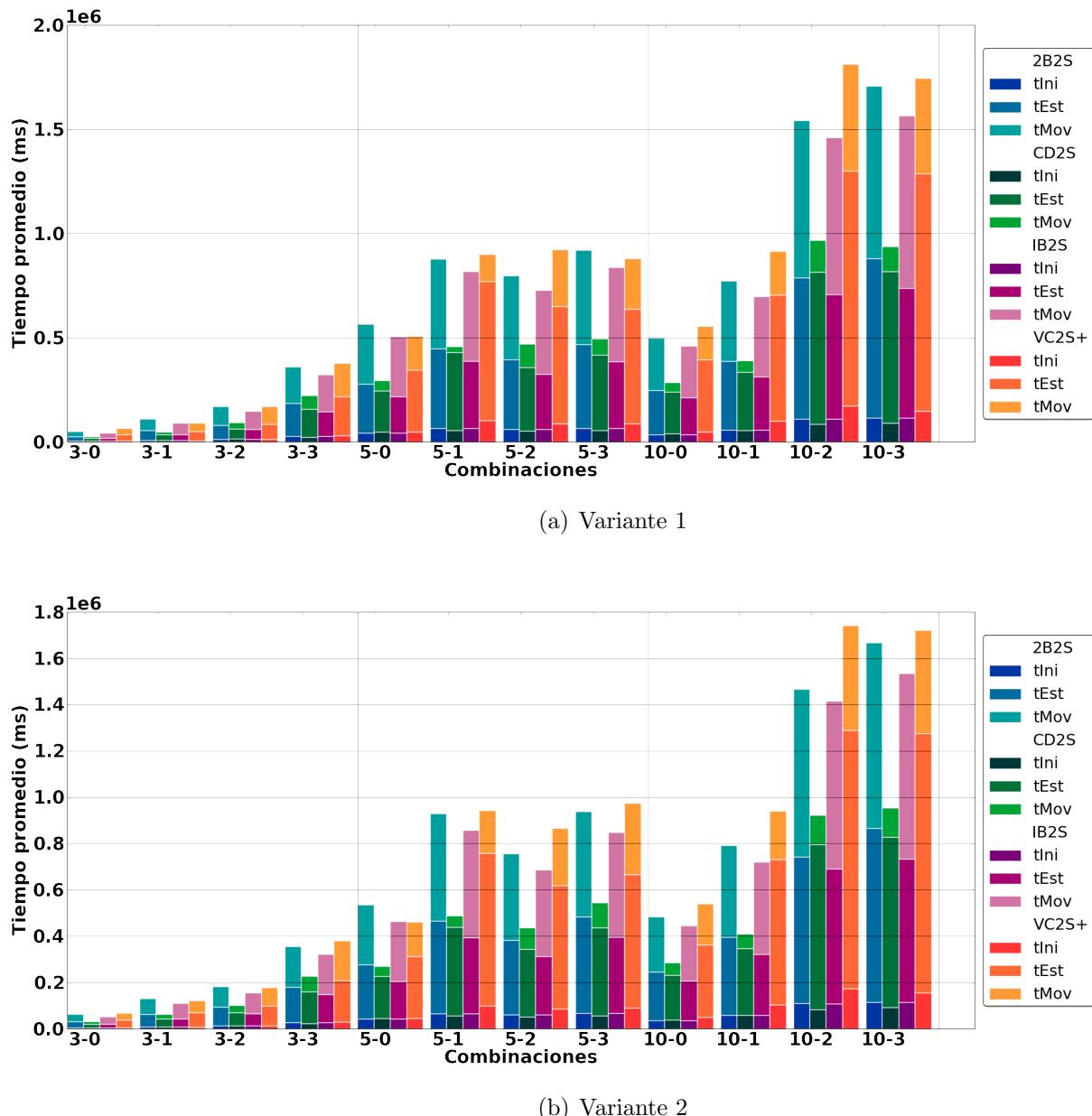


Figura H.1: Tiempo promedio por lote y modo de ejecución

H.1.2 Cardinalidad del *skyline* no espacial

Cabe destacar en la Figura 5.3 p. 95, que para el caso 3-3 donde es igual la cantidad de puntos de referencia que las dimensiones no espaciales, la fracción del skyline no espacial es bastante menor a un 10 %, esto se debe a que las dimensiones no espaciales se generaron de forma independiente entre sí, evitando objetos con valores repetidos en la misma dimensión. Sin embargo, dos o más objetos distintos pueden estar a la misma distancia de un punto de referencia, lo que hace que hayan valores repetidos en las dimensiones espaciales. Más aún podría existir una relación geométrica en el sentido que si dos puntos de referencia están cerca, los objetos con la distancia mínima en esas dimensiones por separado, estarán cerca también. Para mayor detalle, ver el Apéndice G.

H.1.3 Menores y mayores tiempos de ejecución

Adicionalmente, la Figura H.2 muestra los valores de `vecesTMin`, indicando que en un 100 % de los lotes para la variante 1 y un 99,4 % para la variante 2 el algoritmo CD2S tiene el menor tiempo de ejecución entre los demás algoritmos. Esta diferencia de menos de 1 % podría existir por algún caso particular que se haya originado. En segundo lugar se encuentra con un 0,6 % de las veces el algoritmo IB2S. Esto confirma que CD2S no sólo es bueno en cada combinación, sino también con cada caso de prueba.

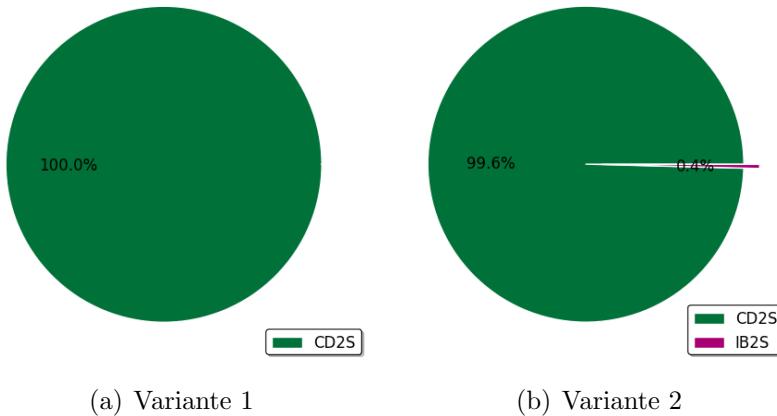


Figura H.2: Porcentaje de lotes en que un algoritmo es el mejor, `vecesTMin`

En cambio, en la Figura H.3 que muestra los valores de `vecesTMax`, el algoritmo VC2S+ domina con el mayor tiempo de ejecución con un 56,9 % y un 58,3 % para las variantes 1

y 2, quedando en segundo lugar el algoritmo 2B2S con un 42,9% y 41,5% de las veces, concluyendo con estas gráficas que estos algoritmos tienen el peor desempeño en tiempo para el experimento realizado.

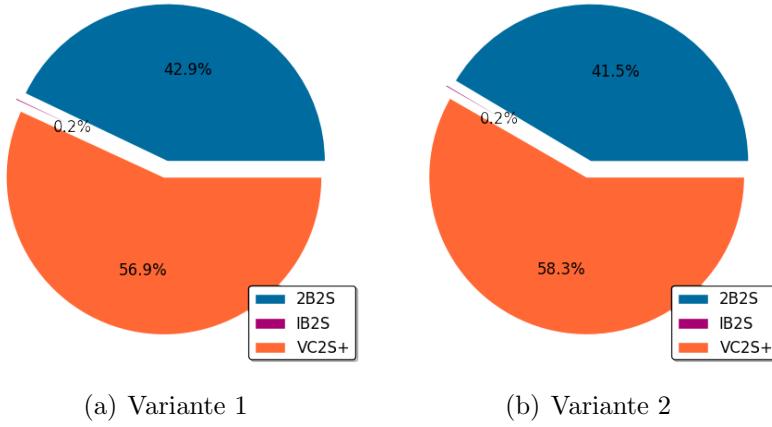


Figura H.3: Porcentaje de lotes en que un algoritmo es el peor, `vecesTMax`

H.1.4 Cambios en el *skyline*

Para actualizar o generar el conjunto *skyline*, es necesario agregar y/o eliminar objetos a la ventana de elementos no dominados, ya que algunos considerados buenos en un principio pueden entrar a la ventana, pero al final de la revisión podrían ser reemplazados por otro mejor. En la Figura H.4 se puede observar la cantidad de cambios que realizaron los algoritmos en promedio, donde el eje *x* indica las combinaciones y el eje *y* el `#cambios realizados`. Debido a que los algoritmos 2B2S y IB2S no restringen la búsqueda de candidatos al *skyline* y que IB2S sólo en el *modo actualización de estatus* aprovecha el *skyline* de la simulación anterior para actualizarlo, ambos necesitan hacer una mayor cantidad de cambios que los demás algoritmos.

La mayor parte de los cambios para los algoritmos 2B2S y IB2S están etiquetados como `+s` que conforman las veces que un elemento es agregado al *skyline* y en un menor porcentaje se realizan los cambios `-s`. Esto se debe a que sólo IB2S en el *modo actualización de estatus* actualiza el *skyline* y a la relación geométrica que existe entre los objetos que aunque para estos algoritmos es necesario evaluar todos los objetos disponibles entre sí para crear el *skyline*, se realiza iterando la matriz secuencialmente y esta estructura provee un cierto

orden de los objetos por su localización, haciendo que se agreguen a la ventana elementos que probablemente conforman el *skyline*.

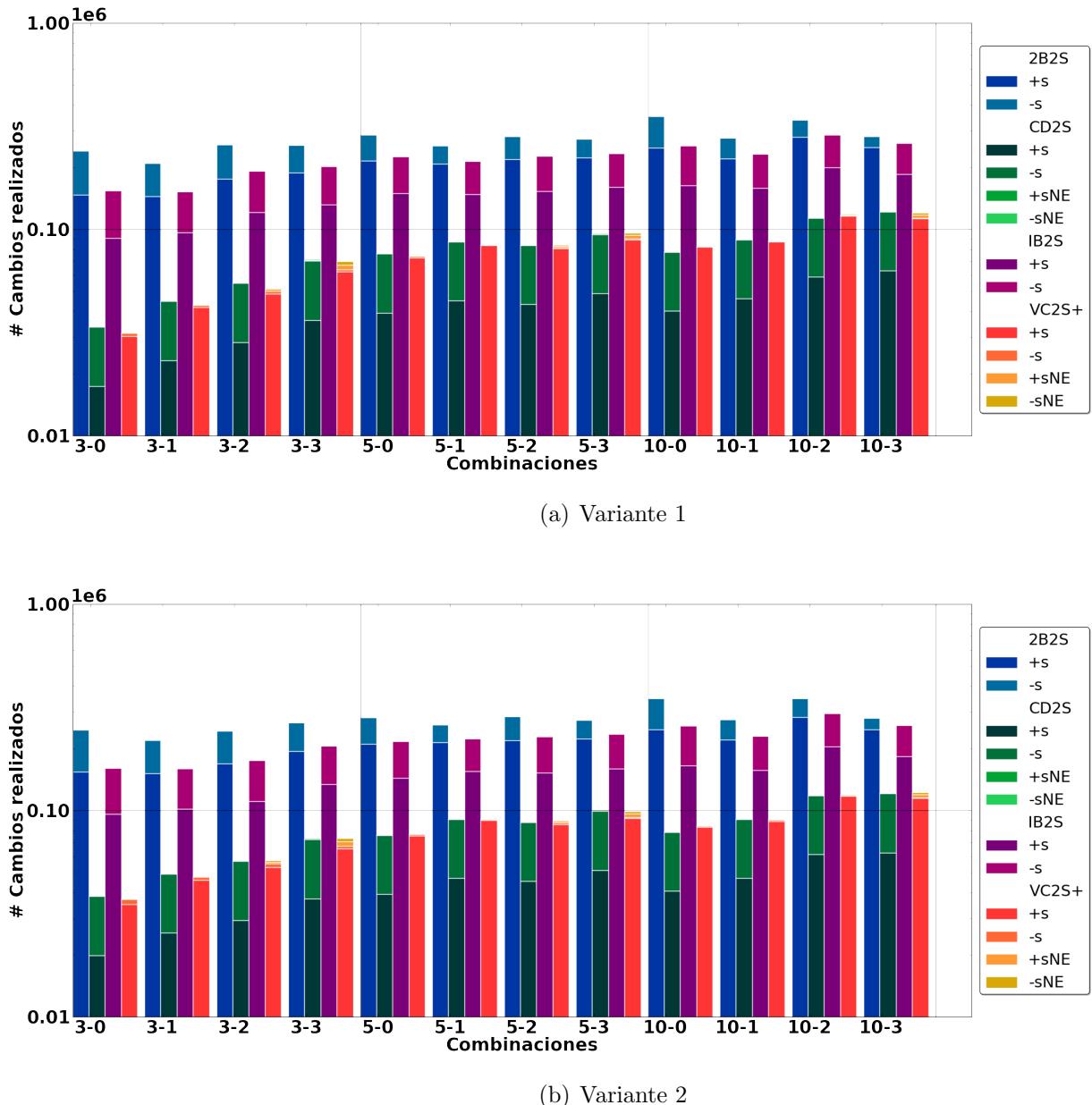


Figura H.4: Cambios realizados en el *skyline* en promedio por lote (Escala logarítmica)

Los algoritmos CD2S y VC2S+, realizan un número similar de cambios en el *skyline*, siendo menor para este último algoritmo, ya que a diferencia de CD2S sólo inserta en la ventana si está seguro que el objeto no es dominado por otro. Por lo tanto, los cambios etiquetados como *-s* para el algoritmo VC2S+ corresponden únicamente a los elementos que deben ser

eliminados por un cambio de posición del punto de referencia móvil.

La fracción de elementos agregados al *skyline* es menor para CD2S que para VC2S+ porque este último no actualiza el *skyline* en algunos patrones de desplazamiento y en el *modo actualización de estatus*, requiriendo la eliminación del conjunto y creación desde el inicio. En una menor proporción se realizan los cambios etiquetados como +sNE y -sNE por ser baja la cardinalidad del *skyline* no espacial.

H.2 Experimento II

H.2.1 Menores y mayores tiempos de ejecución

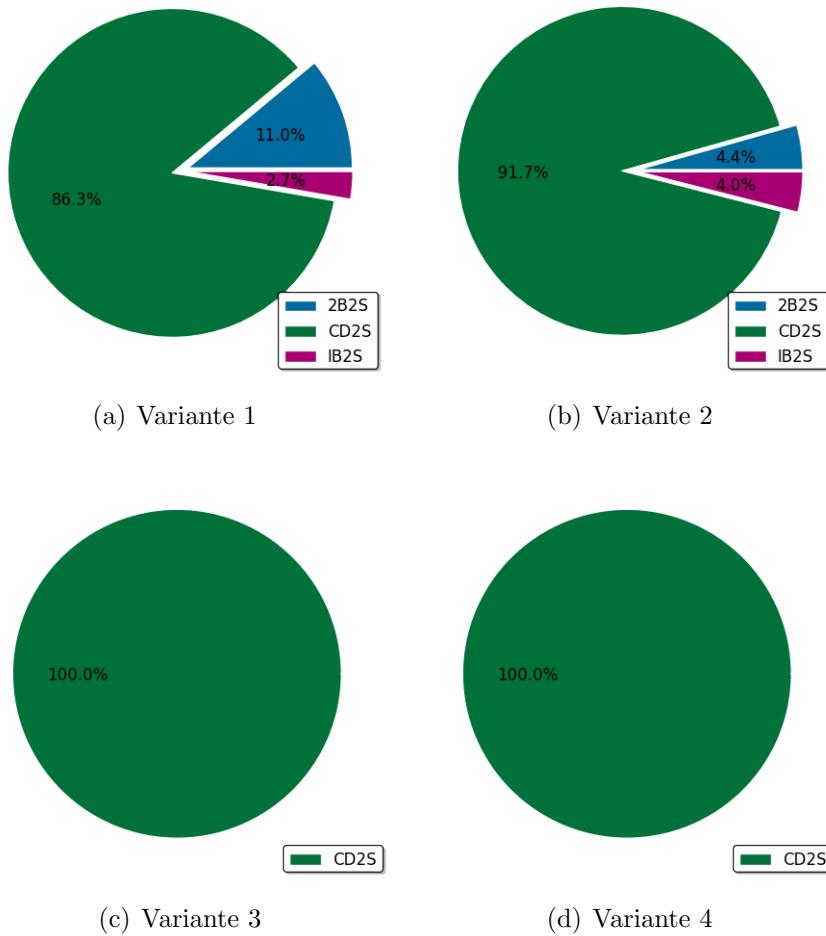


Figura H.5: Porcentaje de lotes en que un algoritmo es el mejor, `vecesTMin`

Al existir un *convex hull* inicial pequeño en relación al plano $CHMax = 10\%$, podrían presentarse casos donde se estén subutilizando las estructuras y procedimientos con una pe-

queña cantidad de candidatos. Esto se observa en la Figura H.5 en la gráfica de `vecesTmin` para la variante 1 que en un 11 % de los lotes ejecutados fue mejor el desempeño del algoritmo 2B2S y en un 2,7 % IB2S indicando que en un porcentaje considerable es mejor calcular desde el inicio el *skyline*. Este porcentaje baja para la variante 2 a un 4,4 % y 4 % respectivamente.

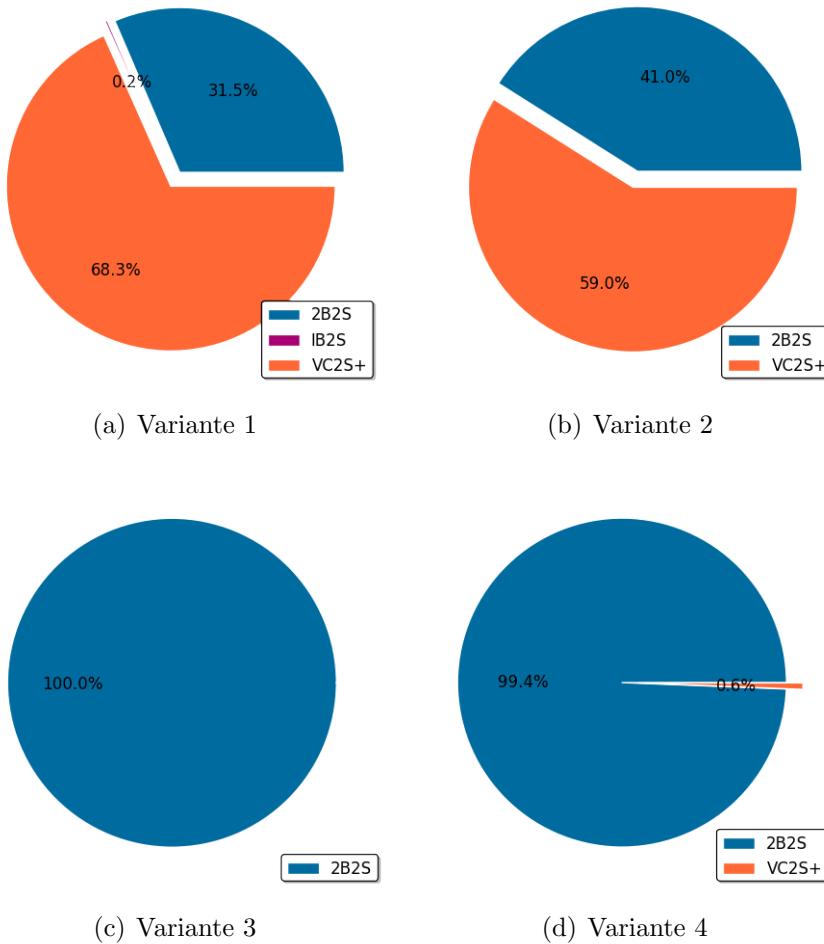


Figura H.6: Porcentaje de lotes en que un algoritmo es el peor, `vecesTMax`

Se concluye con las gráficas de las variantes 3 y 4 que para este experimento el mejor punto para utilizar el algoritmo CD2S es con $CHMax = 50\%$ ya que en un 100 % de los lotes tuvo el menor tiempo.

Este comportamiento también se refleja para VC2S+, en la Figura H.6 de las gráficas de `vecesTmax`, se observa que para la variante 1 en un 68,3 % de los lotes ejecutados, posee el

peor tiempo, mejorando a un 59 % para la variante 2; resultado que cambia drásticamente a un 0 % y 0,6 % para las variantes 3 y 4.

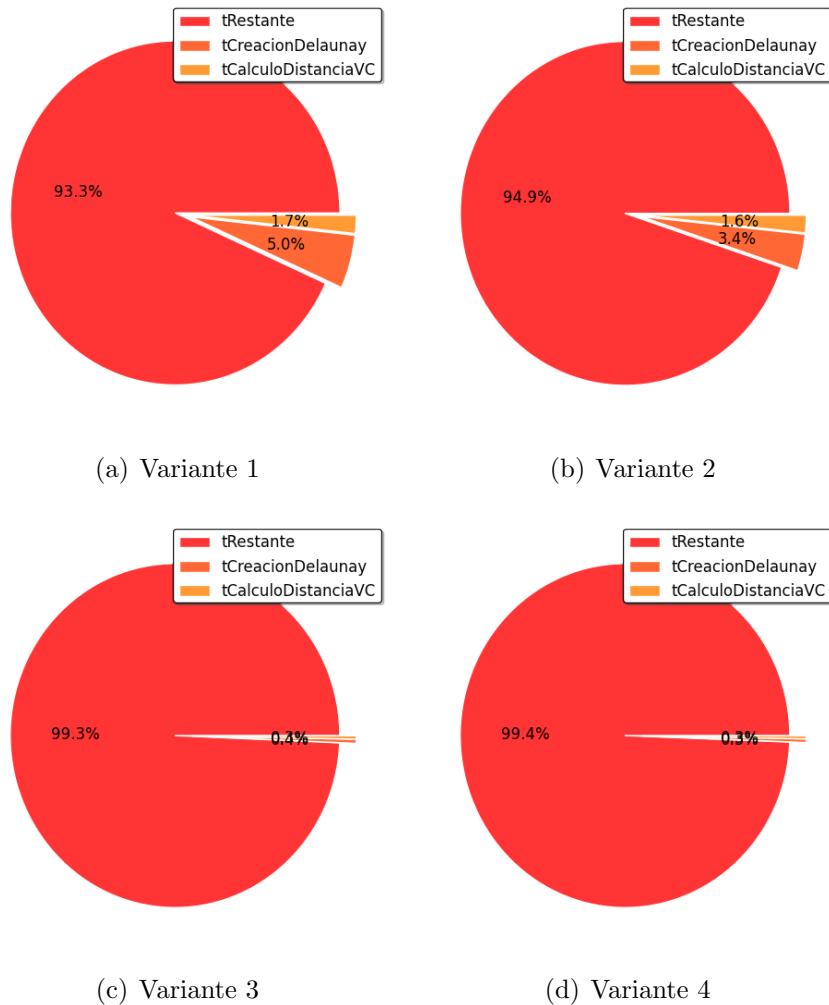


Figura H.7: Distribución del tiempo promedio de ejecución por lote para VC2S+

El motivo de este cambio se debe al tiempo empleado por este algoritmo para la creación del grafo *Delaunay*. Este tiempo *tCreacionDelaunay*, en la Figura H.7 de los tiempos de VC2S+, es de un 5 % del tiempo promedio y un 1,7 % del tiempo es empleado en el cálculo de la menor distancia de la celda de Voronoi de cada objeto a cada punto de referencia *tCalculoDistanciaVC*, cuyo procedimiento sólo se realiza cuando la cantidad de dimensiones no espaciales es igual a 0, siendo ya despreciable para las variantes 3 y 4. Por lo que para este algoritmo al igual que CD2S, en el estudio de $CHMax = 50\%$ es cuando mejor se aprovechan las estructuras y métodos que utiliza el algoritmo en la obtención del *skyline*.

H.2.2 Histograma

Para un estudio de los tiempos de ejecución individuales para cada caso de prueba empleado, se realizaron 3 histogramas visibles en la Figura H.8, los cuales contienen los **tiempos por lotes** de cada uno de los lotes y con cada uno de los algoritmos. Se muestran los **tiempos por lotes** correspondientes al estudio del comportamiento de los algoritmos para $CHMax = 10\%$ y $CHMax = 50\%$, contrastados con los resultados del Experimento I donde el *convex hull* inicial no está limitado.

En el eje *x* de los histograma están los **tiempos por lotes** y en el eje *y* la frecuencia. Los histogramas se realizaron fijando en 10 el número de intervalos, por lo que el grosor de la barra está directamente relacionado con la amplitud del rango de los tiempos de ejecución. Los casos en donde no aparecen las 10 barras de los tiempos para un algoritmo, como ocurre con el algoritmo 2B2S para $CHMax = 10\%$ que sólo aparecen 4 barras, es porque las 6 restantes mas allá de $5,83 \times 10^4$ agrupan a una cantidad poco significativa de lotes, menos del 1 %.

Como se puede observar en todos los histogramas, el algoritmo CD2S es el que posee la mayor frecuencia en los menores tiempos de ejecución, ubicando más del 80 % de los resultados (primeras 2 barras) para $CHMax = 10\%$ entre $[0ms; 0,77 \times 10^4 ms]$, más del 73 % (primeras 3 barras) entre $[0ms; 0,11 \times 10^6 ms]$ para $CHMax = 50\%$ y más del 80 % (primeras 3 barras) entre $[0ms; 0,65 \times 10^6 ms]$ para $CHMax = 100\%$.

En las estadísticas de los histogramas por algoritmo y valor del parámetro $CHMax$, en la Tabla H.1, también se refleja que la mediana M_e y la moda M_o del algoritmo CD2S son las menores con respecto al resto de los algoritmos, situándose la moda para $CHMax = 50\%$, en $M_o = 29.392ms$. La diferencia de la moda del algoritmo CD2S con la siguiente menor moda es de un 80 % y corresponde al algoritmo VC2S+ $M_o = 53.556ms$, siendo más alta la frecuencia absoluta de la clase modal f_{M_o} del algoritmo CD2S, indicando que la moda se repite 327 veces contra 281 veces, lo cual representa a un 34 % de los resultados de la configuración 3.

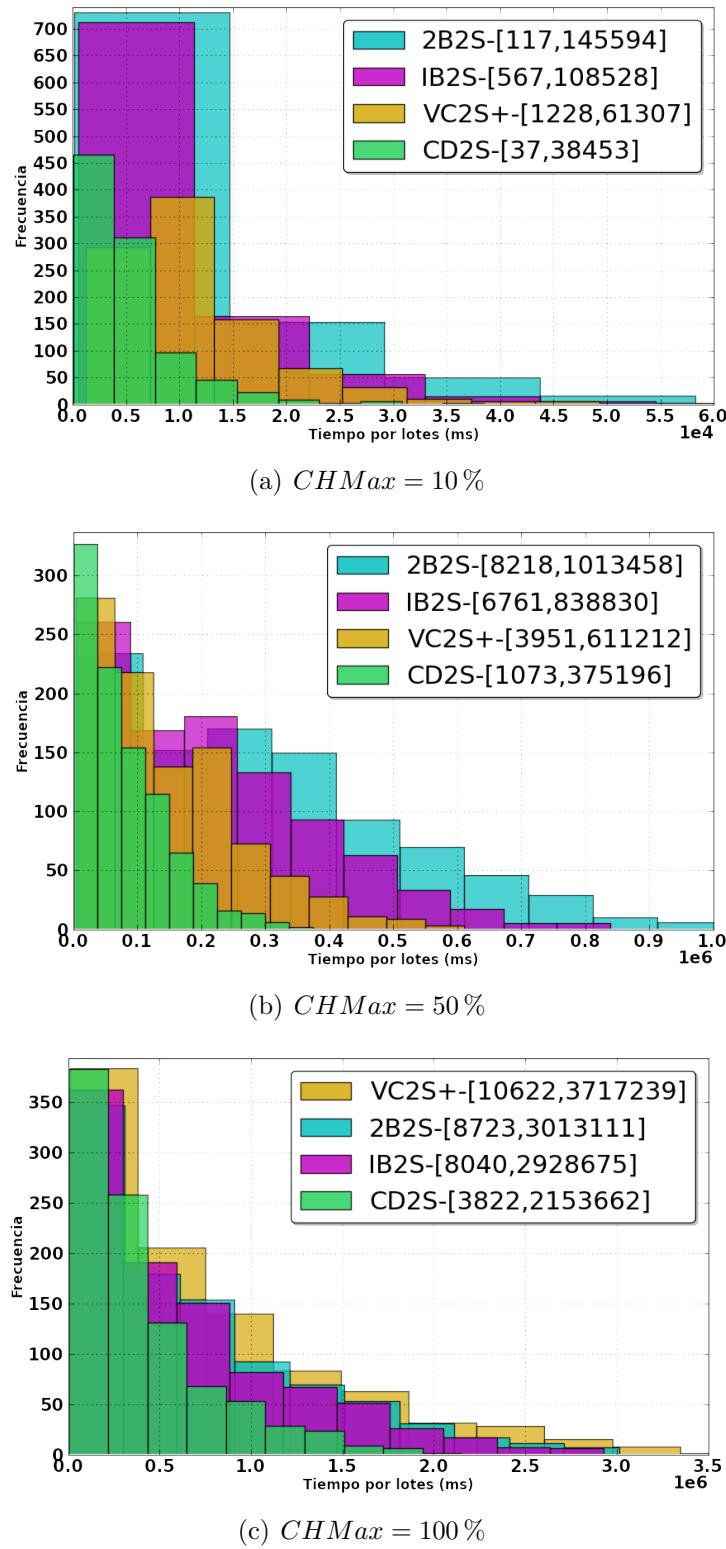


Figura H.8: Histogramas del tiempo de ejecución (Muestra=480 lotes, Intervalos=10)

Algoritmo	CHMax=10 %			CHMax=50 %			CHMax=100 %		
	M_e	M_o	f_{M_o}	M_e	M_o	f_{M_o}	M_e	M_o	f_{M_o}
(1) 2B2S	7.277	8.247	731	264.642	82.657	234	545.504	211.155	347
(2) IB2S	6.071	6.667	713	196.072	68.282	261	489.131	206.402	362
(3) VC2S+	9.464	8.892	386	120.078	53.556	281	551.302	263.886	384
(4) CD2S	3.990	2.923	465	63.504	29.392	327	303.083	165.906	383

Tabla H.1: Estadísticas de los histogramas

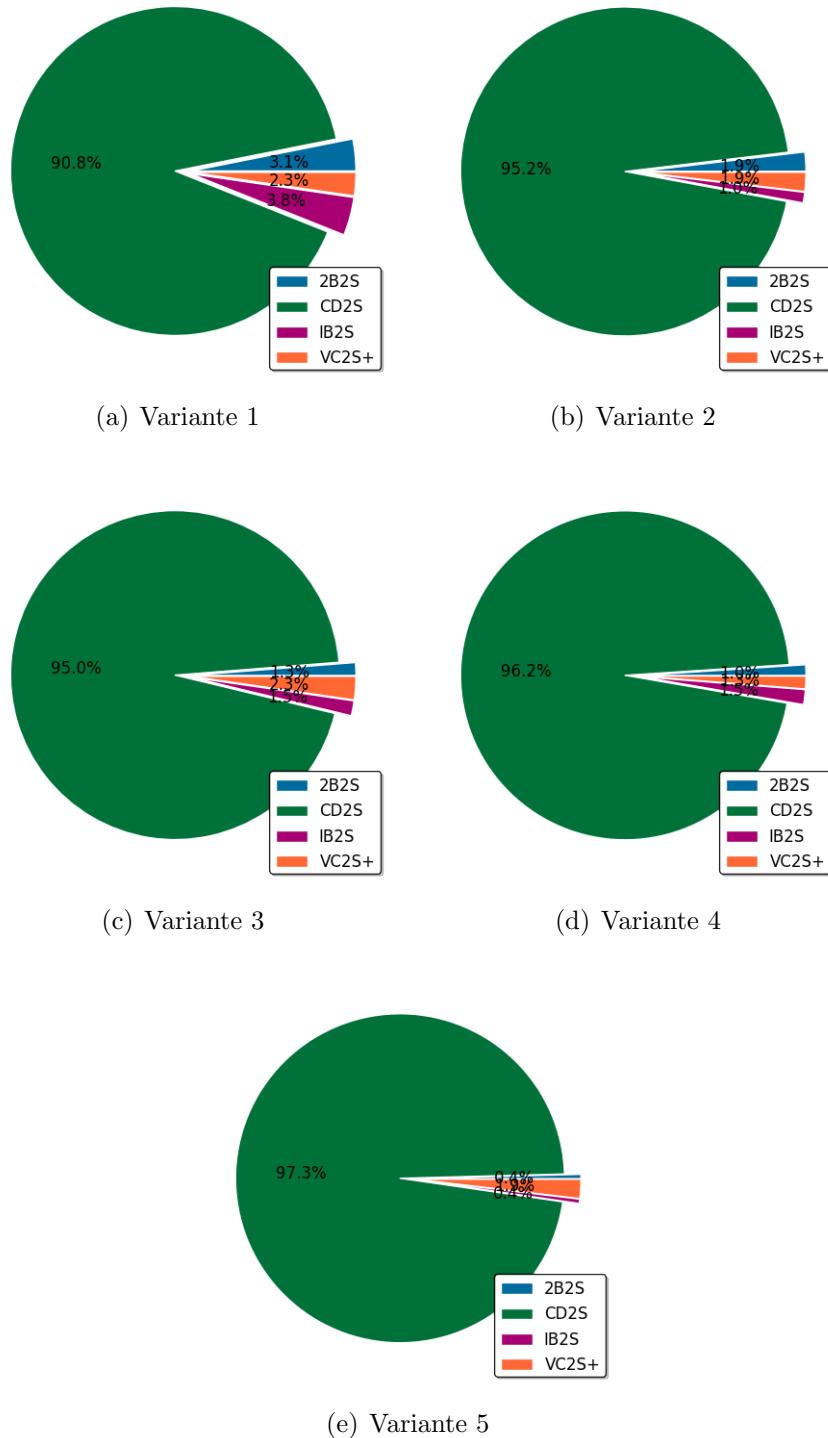
A diferencia de CD2S, el algoritmo 2B2S posee una mayor frecuencia que el resto de los algoritmos, en los tiempos de ejecución mas altos, tanto para $CHMax = 10\%$ como para $CHMax = 50\%$, además que los rangos son más amplios: [117ms; 145.594ms] y [8.218ms;1.013.458ms] en comparación con CD2S cuyos rangos son de [37ms;38.453ms] y [1.073ms;375.196ms]. Para $CHMax = 100\%$, el lugar de 2B2S es reemplazado por el VC2S+ con un rango de [10.622ms; 3.717.239ms] en comparación con el rango de CD2S [3.822ms;2.153.662ms].

H.3 Experimento III

H.3.1 Menores y mayores tiempos de ejecución

Al observar la Figura H.9 del parámetro `vecesTMin`, aumenta la cantidad de veces que el algoritmo CD2S posee el mejor tiempo, de 90,8% en la variante 1 a 97,3% en la variante 5. Aunque en la Figura H.10 del parámetro `vecesTMax`, se reduce el porcentaje de veces en que el algoritmo VC2S+ es peor que el resto de los algoritmos de 52,1% en la variante 1 a un 38,7% en la variante 5, no termina teniendo un lugar significativo en los mejores tiempos de ejecución.

El algoritmo VC2S+ en las combinaciones para 0 dimensiones no espaciales, debería tener un mejor desempeño que los demás, debido a que el algoritmo en su forma original acepta estos casos de prueba y poda el espacio de búsqueda si el vecino del vecino es dominado por otro elemento del *skyline* o por la pila de candidatos a *skyline*. Observando las Figuras 5.7 de tiempo promedio y H.12 de comparaciones efectuadas, la linea correspondiente al algoritmo tanto en el tiempo de ejecución como en el número de comparaciones, tiene sus puntos mínimos en las combinaciones 3 – 0, 5 – 0 y 10 – 0 estando a la par que CD2S.

Figura H.9: Porcentaje de lotes en que un algoritmo es el mejor, `vecesTMin`

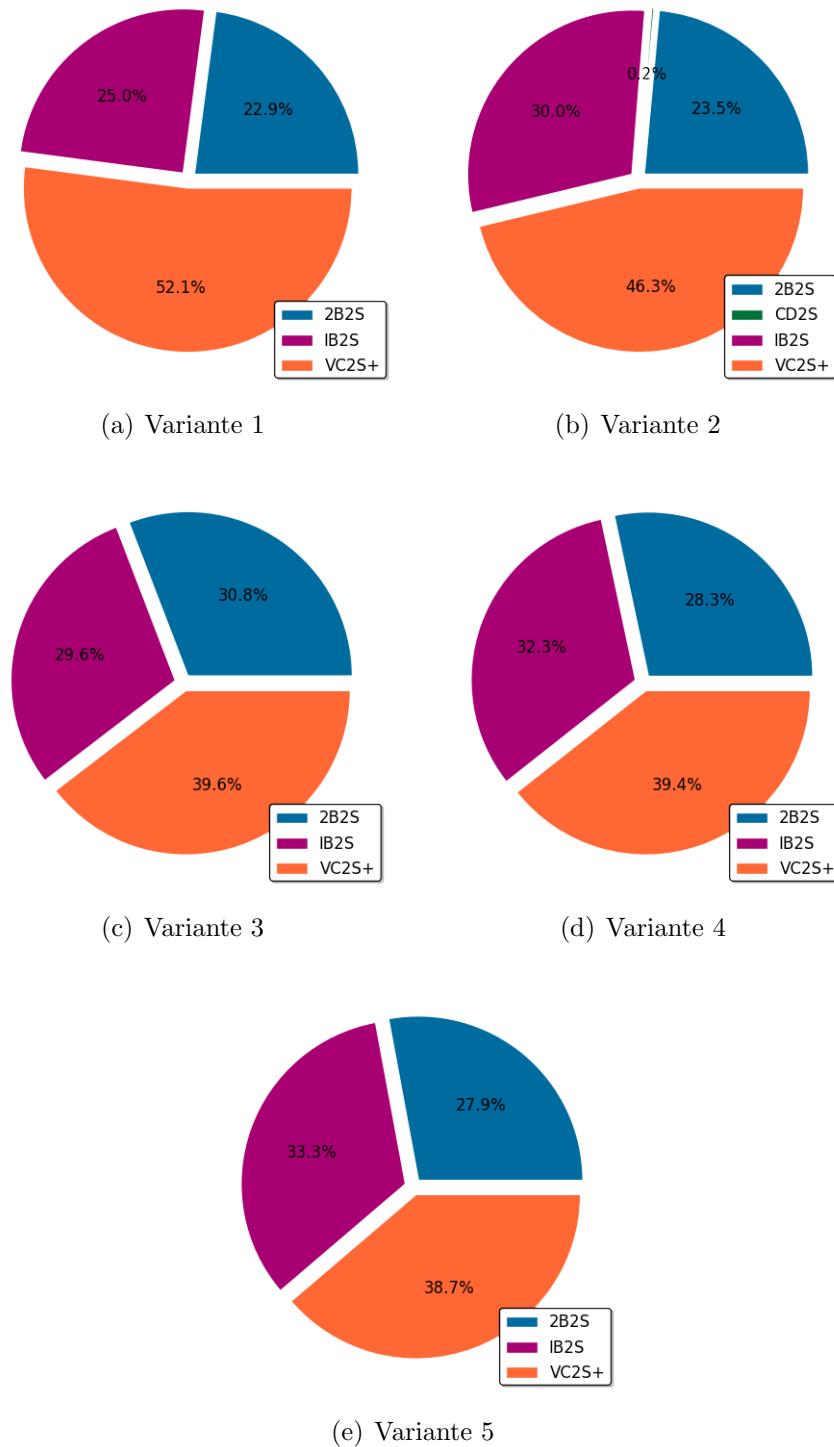


Figura H.10: Porcentaje de lotes en que un algoritmo es el peor, `vecesTMax`

Sin embargo, en el detalle de este experimento en cada combinación en el Apéndice E, no se observa un menor **tiempo promedio** que CD2S, ni siquiera un menor tiempo en el **mejor tiempo** realizado por cada algoritmo. El tiempo promedio más cercano es para la variante 1 en la combinación 5(5-0) donde el algoritmo CD2S se demora $3.039\ ms$ y el VC2S+ $3.668\ ms$, siendo en sólo 6 de los 40 lotes para esta combinación el algoritmo con el menor tiempo **vecesTMin**. En contraste, el número de veces en que el algoritmo tiene el mayor tiempo **vecesTMax**, se reduce significativamente en estas combinaciones, siendo 0 en la combinación 9 (10-0) para las variantes 1,2 y 3.

H.3.2 Candidatos en el *skyline*

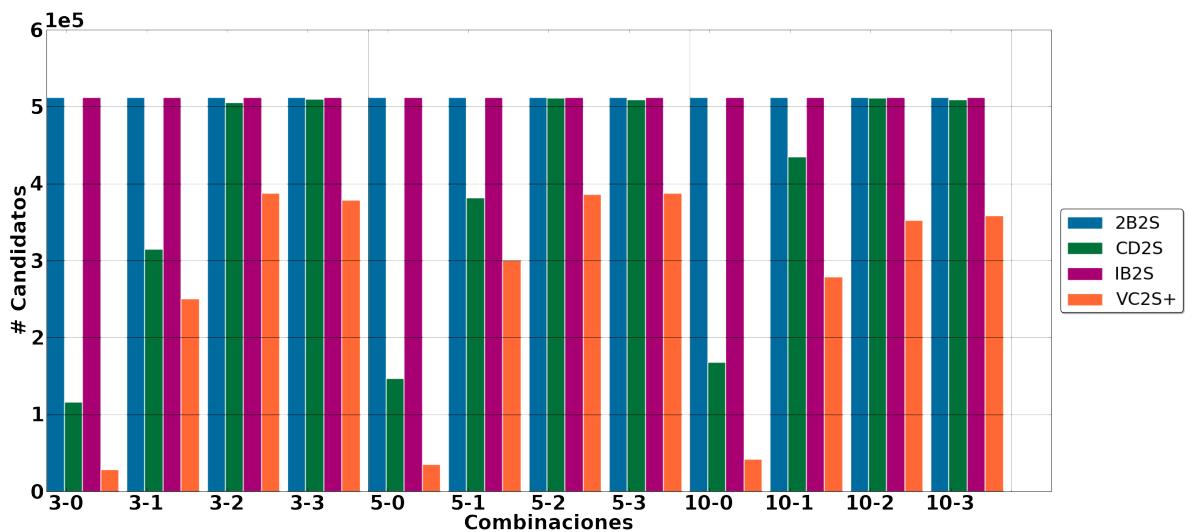


Figura H.11: Candidatos al *skyline* en promedio de todas las variantes por lote

Una consecuencia del aumento del valor de *DistMax* ocurre en la Figura H.11 de candidatos al *skyline*, donde: incrementa la cardinalidad del *skyline* de 33.877 objetos en la variante 1 a 50.121 objetos en la variante 5; se reduce la fracción del *skyline* estático de 37,8% en la variante 1 a 25,5% en la variante 5; y aumenta el porcentaje del *skyline* dinámico de 15,3% a 27,9%, el cual implica una mayor cantidad de cambios en el *skyline*. Como CD2S no tiene que ser ejecutado en el *modo actualización de estatus*, los objetos del *skyline* estático se obtienen al comienzo y no cambian en las simulaciones consecutivas. Al reducirse este porcentaje se desaprovecha esta propiedad.

H.3.3 Comparaciones de dominancia

Sobre el número de comparaciones promedio (Figura H.12), en 10 de las 15 combinaciones $x = 0$ con $x = 3, 5, 10$ para todas las variantes, VC2S+ si tiene un valor menor que las realizadas por el algoritmo CD2S.

Incluso en la Figura H.11 de candidatos al *skyline*, el número promedio de objetos visitados por VC2S+, en todas las variantes del experimento, es menor a la mitad del tamaño de la lista de candidatos para CD2S para estas combinaciones, pero no parece ser suficiente como para reflejarse en el tiempo de ejecución.

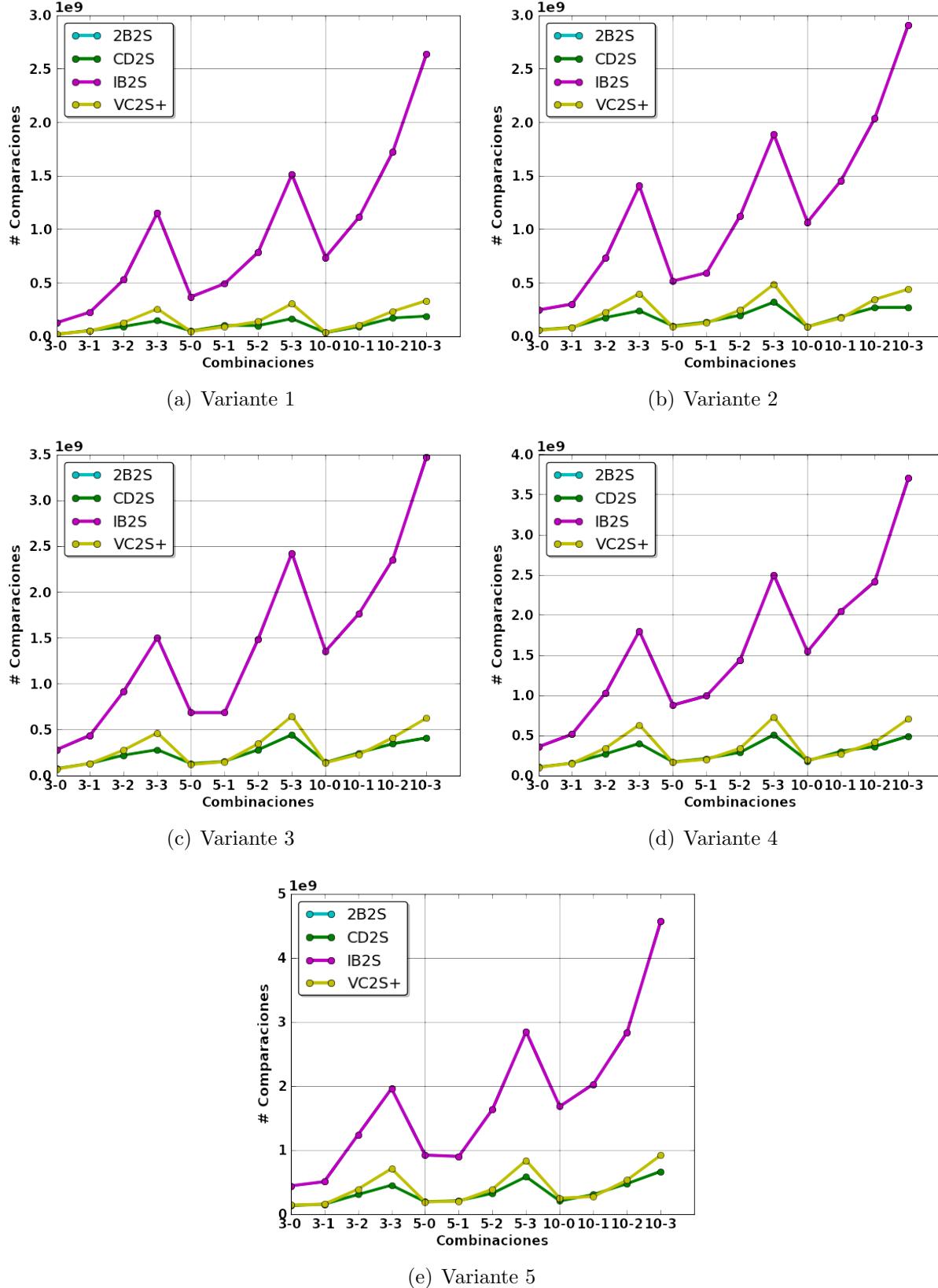
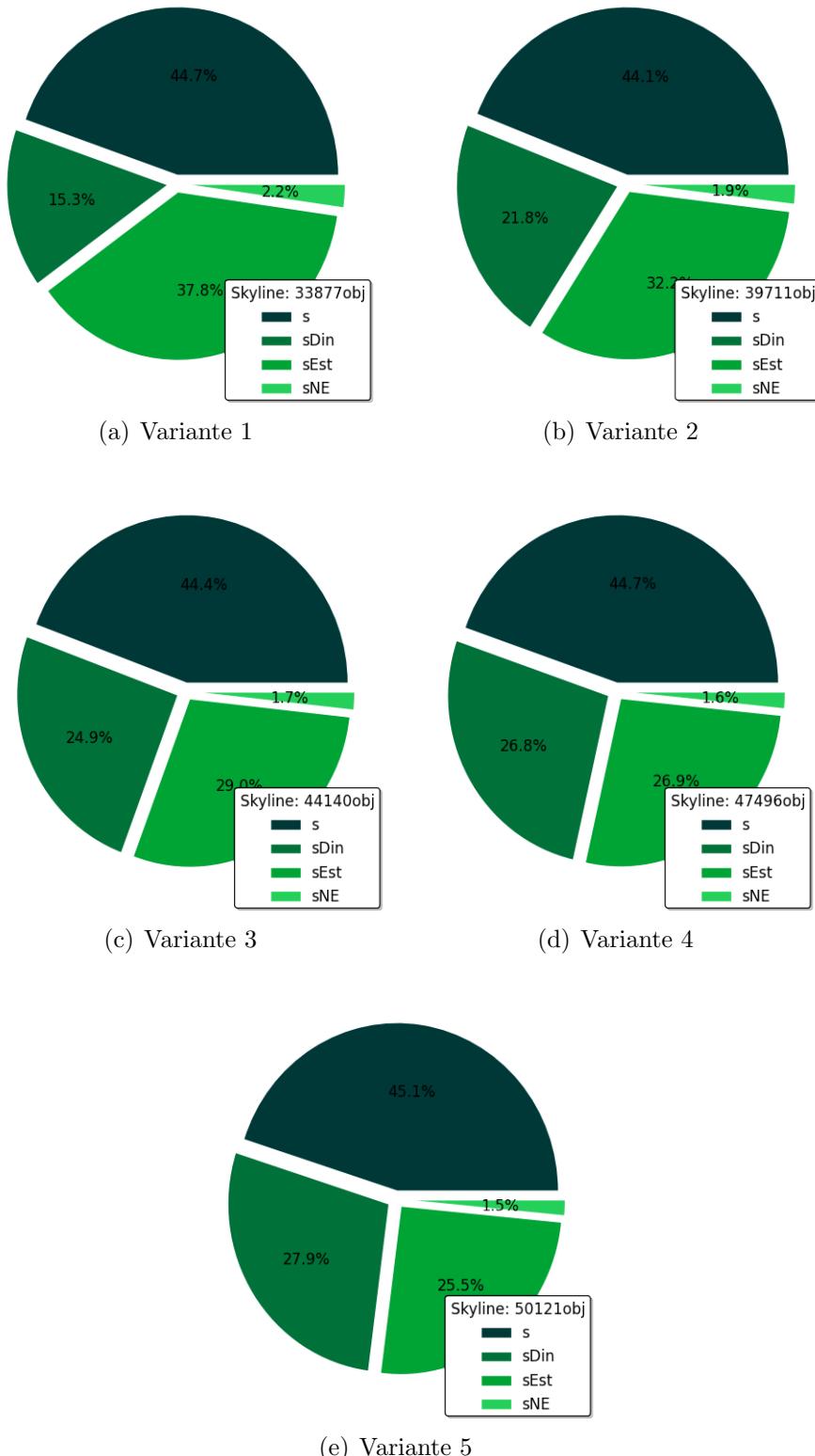


Figura H.12: Comparaciones efectuadas en promedio por lote

Figura H.13: Composición promedio del *skyline* por lote

H.4 Experimento IV

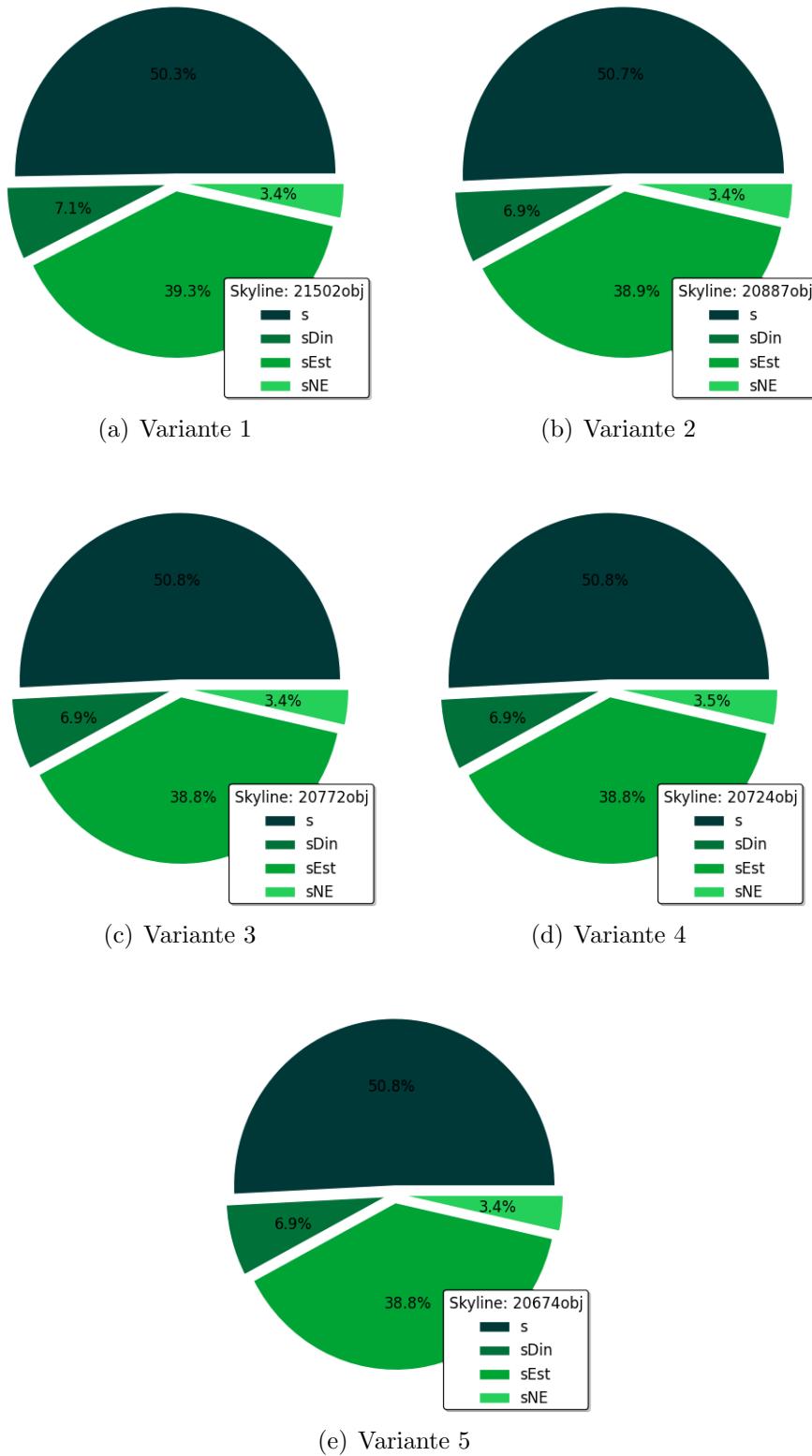
H.4.1 Tiempo y cardinalidad del *skyline*

El *skyline* en la Figura H.14 se mantiene en una baja cardinalidad en todas las variantes. Existen 21.502 objetos para la variante 1 en la Figura H.14(a) contra 33.877 objetos para la variante 1 del Experimento III en la Figura H.13(a) que representa la cardinalidad más baja de ese experimento.

Los algoritmos 2B2S y VC2S+ que no efectúan una actualización del *skyline* en el *modo actualización de estatus* sino que reinician su cálculo en cada simulación, no parecen tener un cambio notable en el tiempo de ejecución.

El algoritmo 2B2S disminuye ligeramente el tiempo desde la variante 1 hasta la variante 5 posiblemente porque la cardinalidad del *skyline* en la Figura H.14 decrece de 21.502 objetos a 20.674 objetos. Efecto contrario para el algoritmo VC2S+ que aumenta ligeramente el tiempo de ejecución, probablemente debido a la actualización que debe realizar en la lista de disponibles para eliminar o agregar los elementos que cambiaron y generar el grafo *Delaunay* correspondiente, así como también debido al cómputo de los nuevos vecinos de Voronoi y al cálculo de las distancias de los objetos que nunca habían estado disponibles y ahora lo están.

Cabe destacar que el tiempo empleado para el algoritmo VC2S+ en calcular las distancias de las celdas de Voronoi a los puntos de referencia `tCalculoDistanciaVC` y en construir el grafo `tCreacionDelaunay` en la Figura H.15 no baja de un 4% y 18% del tiempo promedio de ejecución para este algoritmo en cualquiera de las variantes, indicando que la representación del plano no es la ideal para este experimento.

Figura H.14: Composición promedio del *skyline* por lote

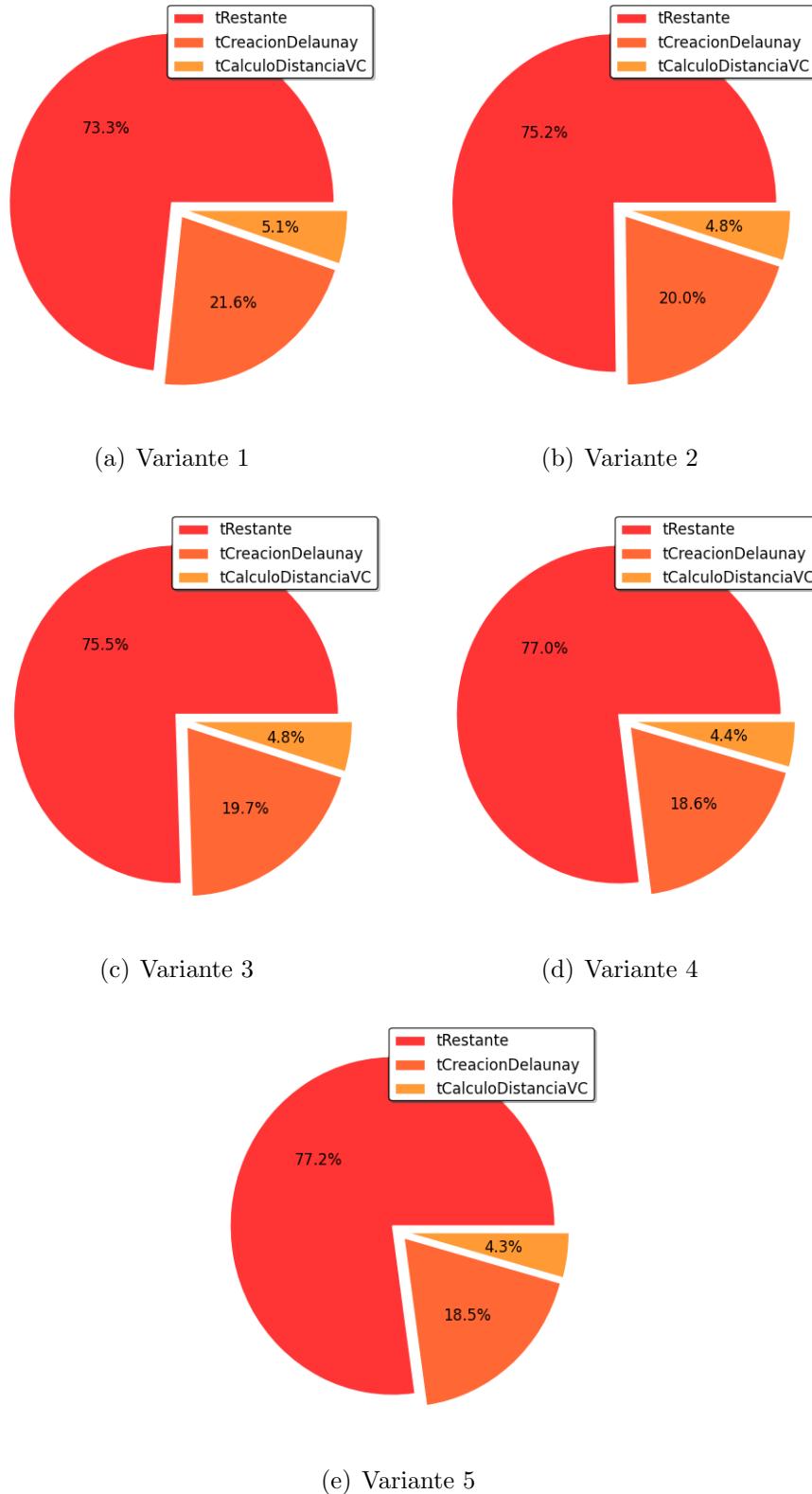


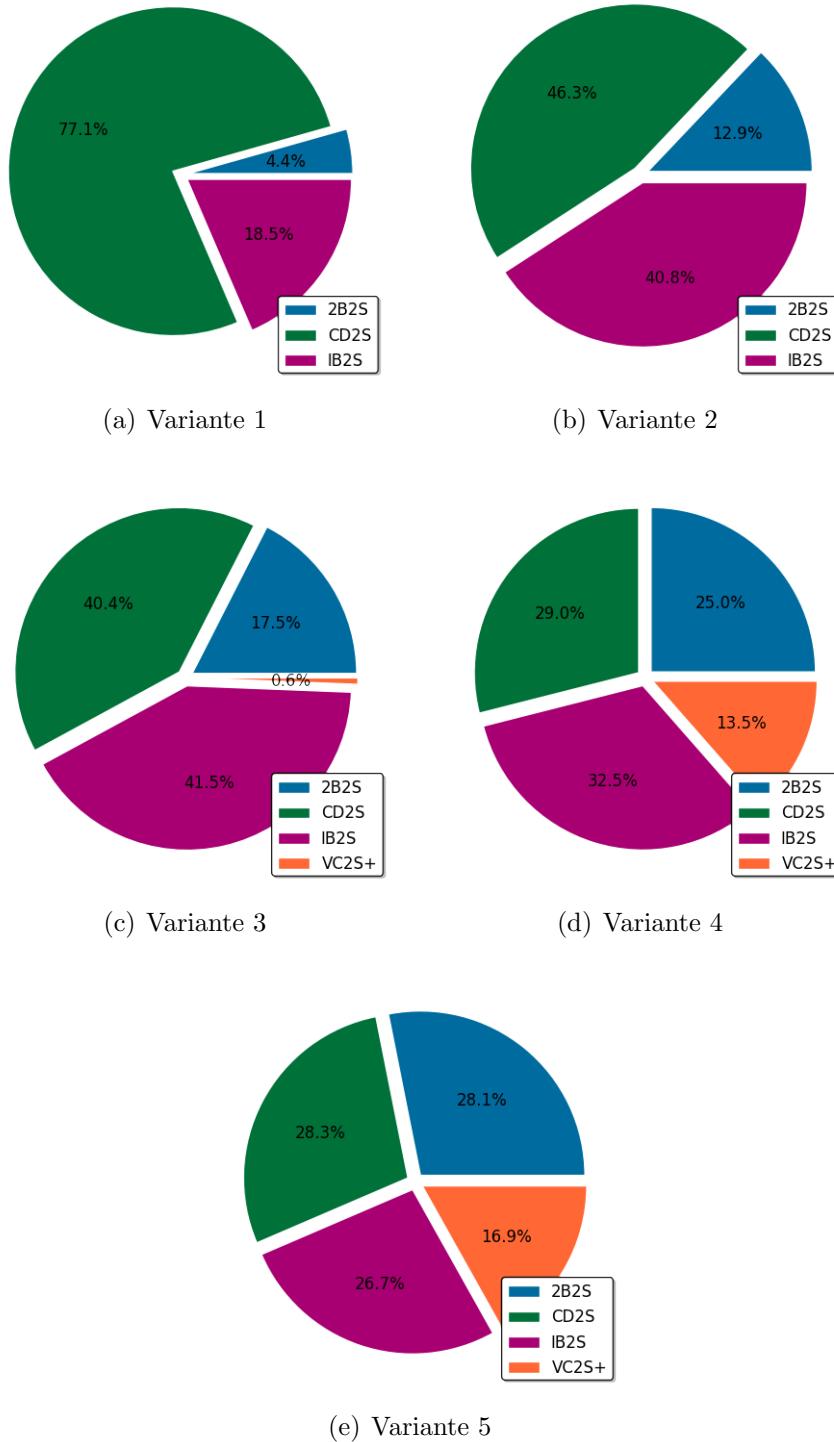
Figura H.15: Distribución del tiempo promedio de ejecución por lote para VC2S+

H.4.2 Menores y mayores tiempos de ejecución

En la Figura H.16 de `vecesTMin`, se observa que los mejores tiempos en la variante 1 están comprendidos por los algoritmos: CD2S con el 77,1 %, IB2S con el 18,5 % y 2B2S con el 4,4 %; cambiando en la variante 5 a: 28,3 %, 28,1 %, 26,7 % y 16,9 % para los algoritmos CD2S, 2B2S, IB2S y VC2S+, lo que indica que en esta variante es bastante similar el resultado de ejecutar un algoritmo u otro.

Así como también se observa la Figura H.17 de `vecesTMax`, que los peores tiempos en la variante 1 están comprendidos por los algoritmos: 2B2S con el 52,9 % y VC2S+ con el 47,1 %; cambiando en la variante 5 a: 48,5 %, 34,8 % y 16,7 % para los algoritmos VC2S+, 2B2S y CD2S.

Si un algoritmo esta listado en las gráficas de `vecesTMin` y `vecesTMax` para una misma variante, significa que sus resultados son muy cambiantes entre sí, pero esto ocurre en 3 de los 4 algoritmos (en las variantes 4 y 5), significando que para estas variantes el desempeño en tiempo de los algoritmos es similar, lo que confirma nuevamente, que disminuyen las diferencias al aumentar el *%Cambio*.

Figura H.16: Porcentaje de lotes en que un algoritmo es el mejor, `vecesTMin`

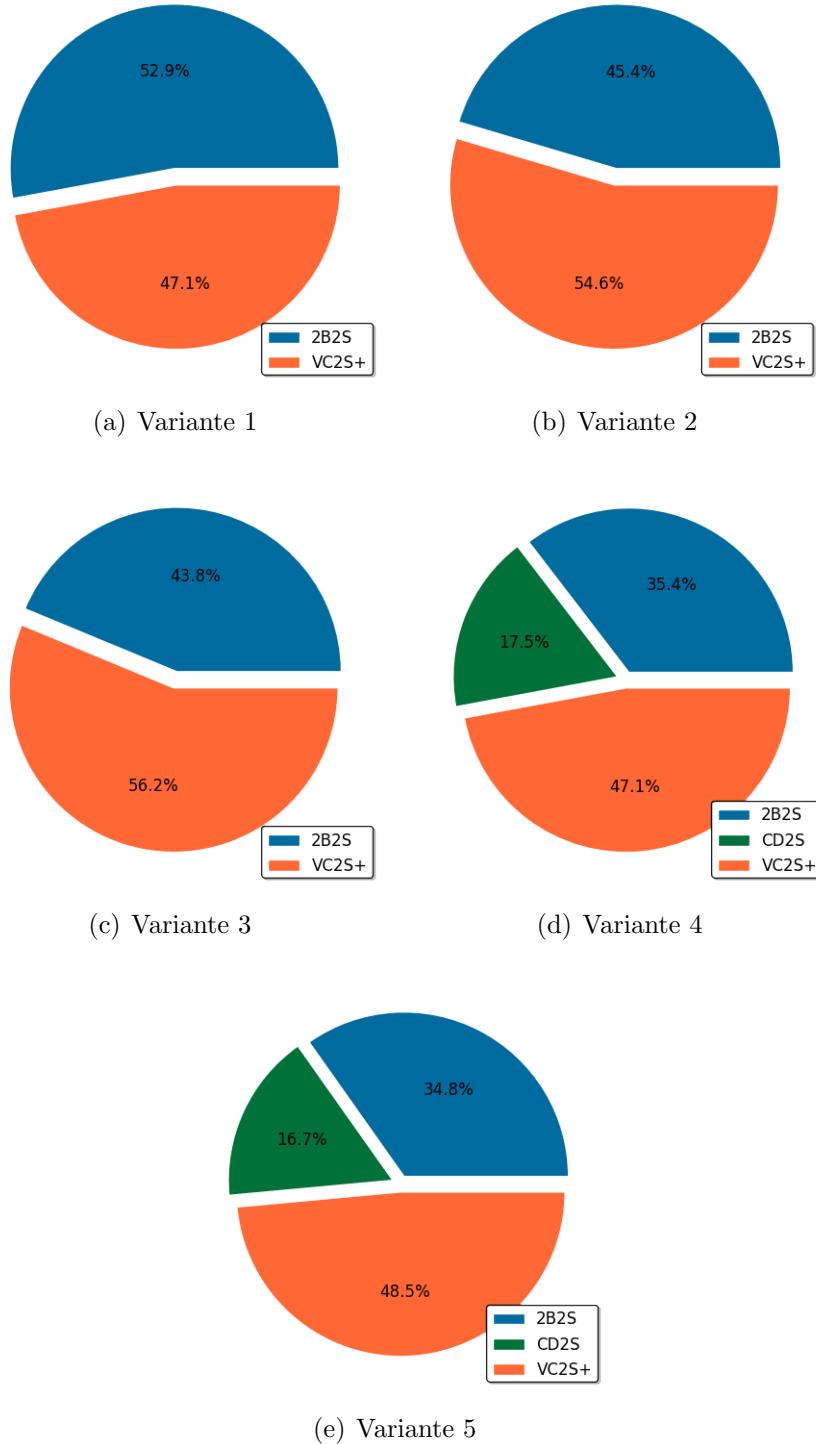


Figura H.17: Porcentaje de lotes en que un algoritmo es el peor, vecesTMax

H.4.3 Candidatos en el *skyline*

A medida que aumenta el número de dimensiones no espaciales, aumenta, en la Figura H.18, la cardinalidad del conjunto de candidatos para CD2S o visitados para VC2S+. Sólo para cero y una dimensión no espacial, los algoritmos tienen una gran diferencia con respecto a 2B2S y IB2S que deben explorar todos los objetos disponibles para realizar las comparaciones de dominancia y actualizar o construir el *skyline* respectivamente.

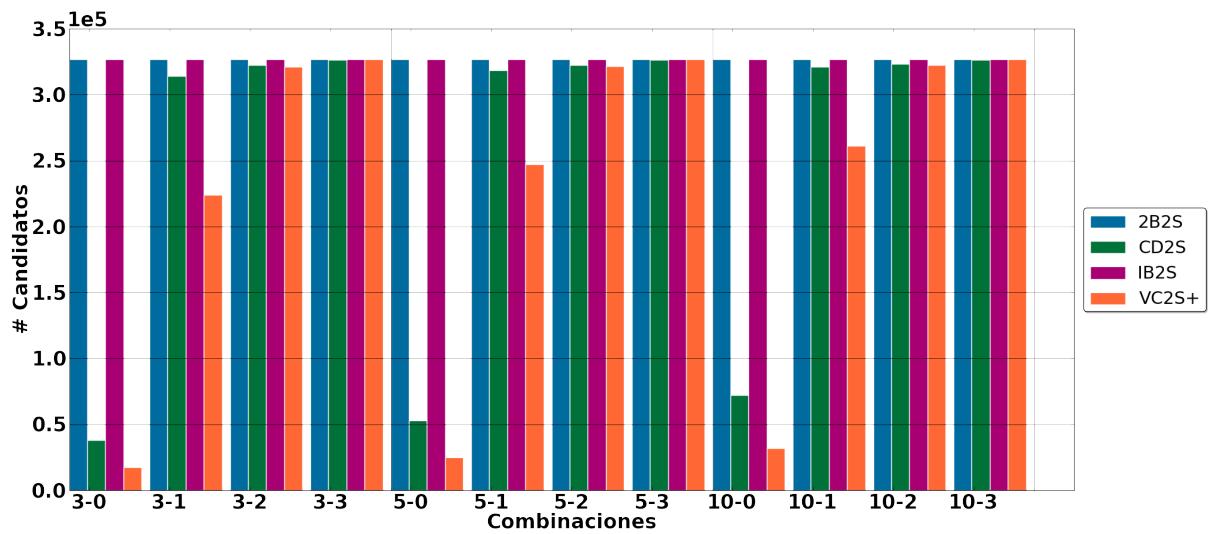


Figura H.18: Candidatos al *skyline* en promedio de todas las variantes por lote

H.4.4 Comparaciones de dominancia

En el número de comparaciones, se observa en las gráficas de comparaciones de la Figura H.19, una mayor estabilidad en los resultados notándose un comportamiento común, estando muy cercanas las comparaciones que realizan los algoritmos VC2S+ y CD2S y más distantes de IB2S y 2B2S. Incluso para las combinaciones 10-3 de la variante 4 y 3-3, 5-3 y 10-3 de la variante 5, el algoritmo VC2S+ ejecuta el menor número de comparaciones.

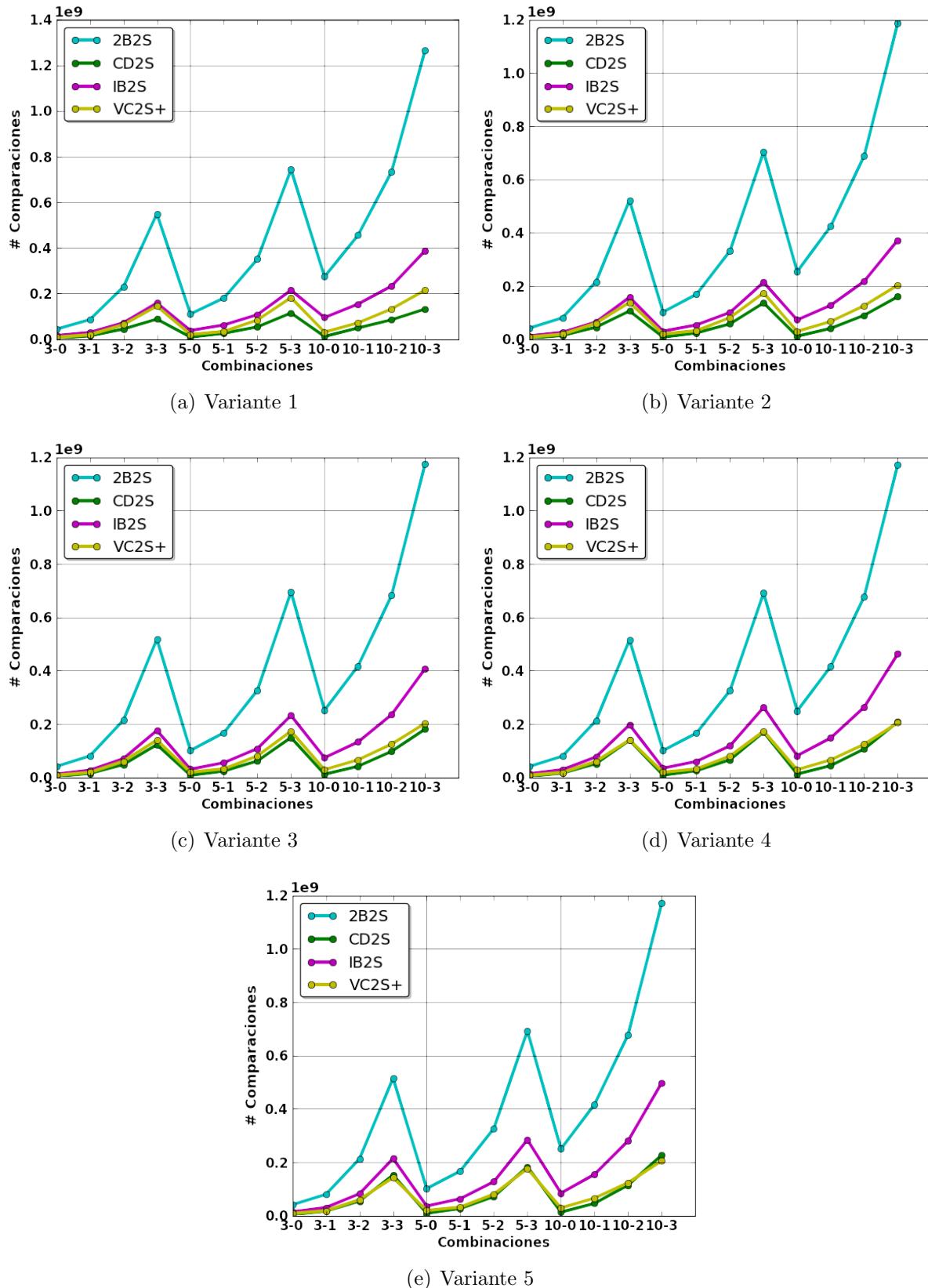


Figura H.19: Comparaciones efectuadas en promedio por lote

H.5 Experimento V

H.5.1 Menores y mayores tiempos de ejecución

Respecto a las veces en que un algoritmo posee el menor tiempo, `vecesTmin`, en la Figura H.20, el algoritmo CD2S domina con un 67,1 % para la variante 1, 97 % para la variante 2 y 95,8 % para la variante 3. En cambio, el número de veces que el algoritmo CD2S posee el peor tiempo, `vecesTmax`, es de 2,1 % en la variante 1 y 0 % en las variantes 2 y 3, lo cual significa que el tamaño del *convex hull* inicial de la variante 1, no compensa a las estructuras y métodos que el algoritmo emplea para actualizar el *skyline*.

Para el algoritmo VC2S+ también se reduce el valor de `vecesTmax`, pasando de un 54,2 % para la variante 1 a 8,7 % para la variante 3, lo que muestra una gran mejoría para este algoritmo, aún cuando tiene una participación menor al 2,5 % en las gráficas de `vecesTmin`.

H.5.2 Histograma

En los histogramas de la Figura H.21, realizados de la misma manera que para el Experimento II, se puede observar por la anchura de los intervalos utilizados para generar los histogramas (fijado en 10) y los rangos de los tiempos registrados para los algoritmos, que en general los tiempos del algoritmo CD2S son mejores que los tiempos de los demás algoritmos.

La **mediana**, M_e , la **moda**, M_o , y la **frecuencia de la moda** f_{M_o} presentes en la Tabla H.2, indican que una gran parte de los resultados del algoritmo CD2S se sitúa en los menores tiempos de ejecución, siendo M_e en la variante 2 363.325 ms en comparación con la peor mediana 1.025.829 ms que corresponde al algoritmo 2B2S. La moda para el algoritmo CD2S en la variante 2 también es bastante menor, ubicando a más de la mitad de los resultados $f_{M_o} = 138$ en la clase modal 275.659 ms, en contraste con el algoritmo 2B2S cuya moda es 726.854 ms con $f_{M_o} = 100$.

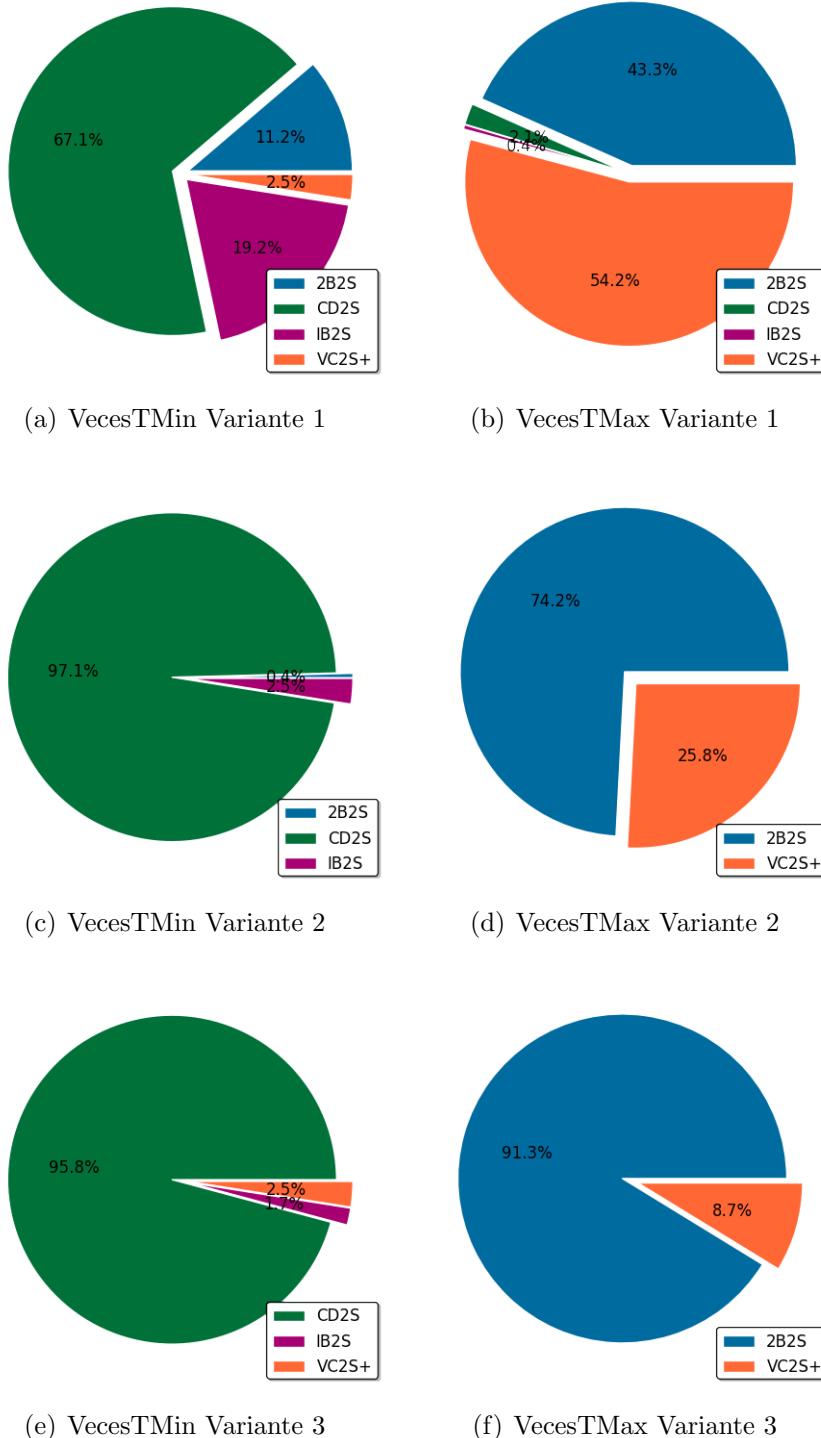


Figura H.20: Porcentaje de lotes en que un algoritmo es el mejor `vecesTMin` o el peor `vecesTMax`

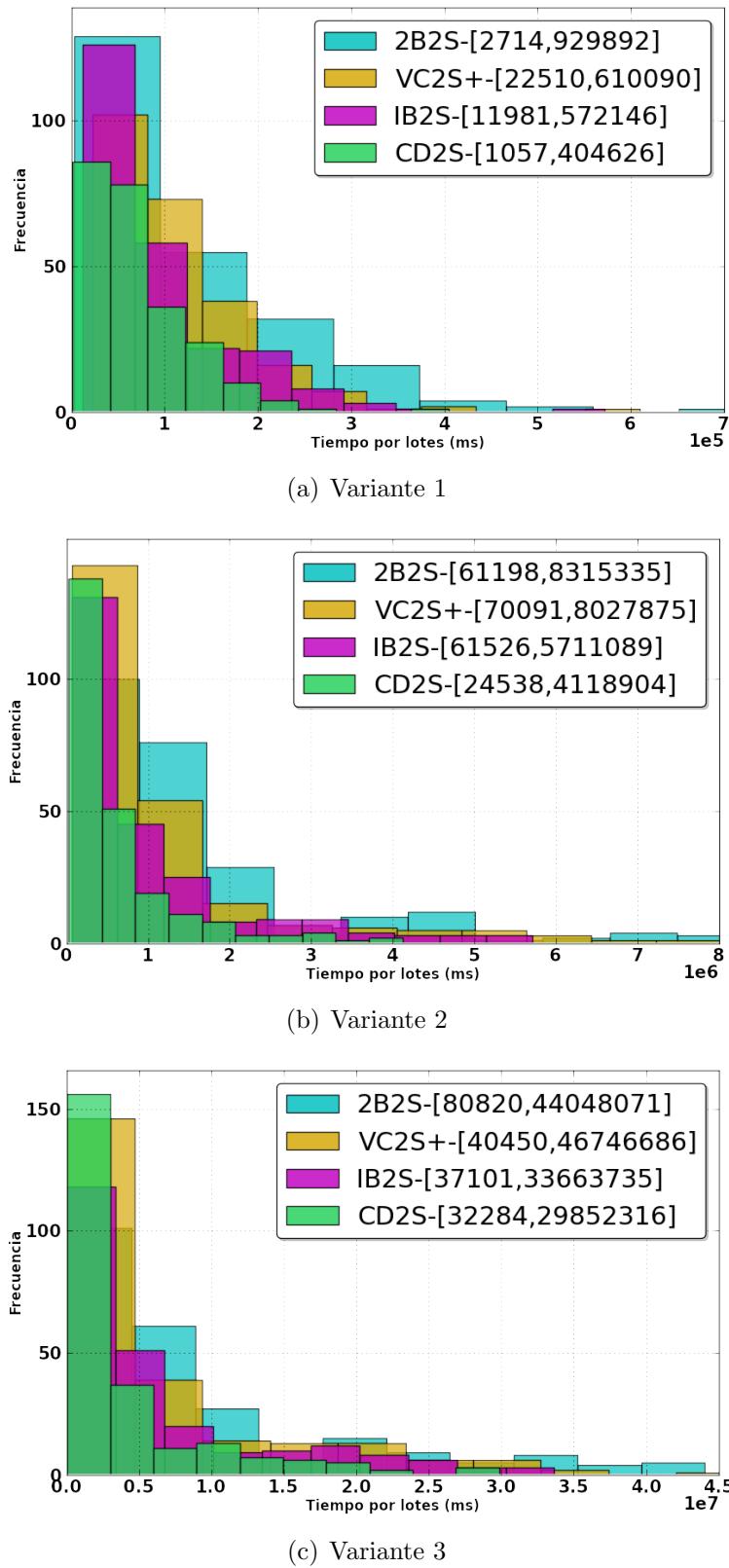


Figura H.21: Histogramas del tiempo de ejecución (Muestra=240 lotes, Intervalos=10)

Algoritmo	CHMax=10 %			CHMax=25 %			CHMax=50 %		
	M_e	M_o	f_{M_o}	M_e	M_o	f_{M_o}	M_e	M_o	f_{M_o}
(1) 2B2S	88.323	61.633	129	1.025.829	726.854	100	5.194.858	3.230.247	101
(2) IB2S	64.759	48.363	126	596.642	402.583	131	3.451.903	2.181.935	118
(3) VC2S+	96.522	68.261	102	730.082	560.592	143	3.308.642	2.735.751	146
(4) CD2S	58.245	37.979	86	367.325	275.659	138	1.988.787	1.723.893	156

Tabla H.2: Estadísticas de los histogramas

H.5.3 Comparaciones de dominancia y cambios en el *skyline*

En la Figura 5.12(b) aumenta la cantidad de comparaciones de dominancia que debe realizar cada algoritmo desde la variante 1 a la 5, así como en la Figura H.22(a), el número de cambios a realizar para retornar el *skyline*, lo cual se debe al aumento de la cardinalidad del *skyline* en la Figura 5.13

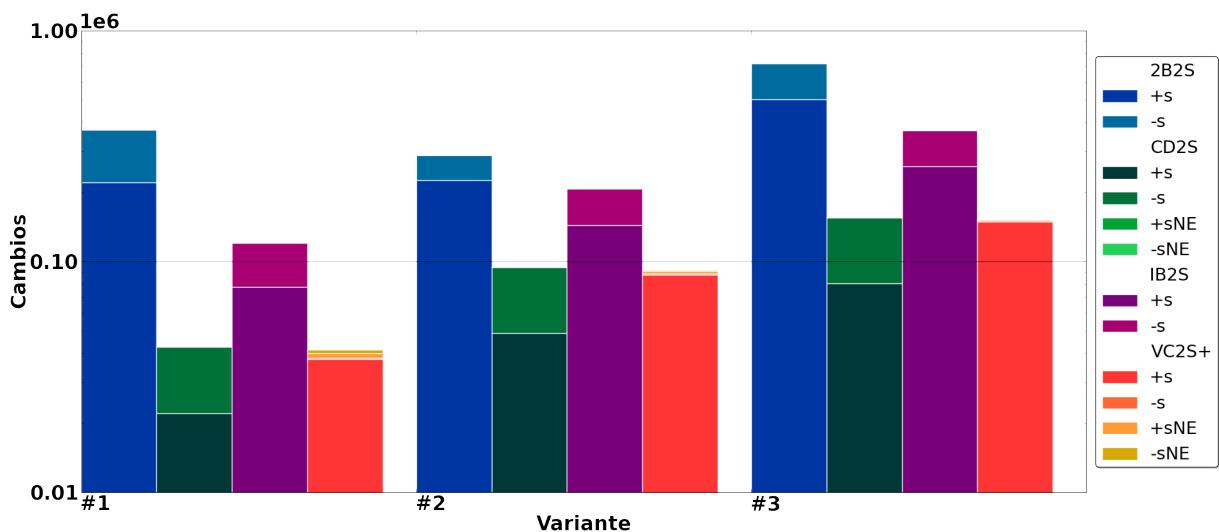
(a) Cambios en el *skyline* en promedio (Escala logarítmica)

Figura H.22: Resultados generales

H.6 Experimento VI

H.6.1 Menores y mayores tiempos de ejecución

Por lo que respecta al valor de `vecesTmin`, en las Figuras H.23(a) y H.23(b), se observa que es igual al 100 % y 93,8 % de los lotes para el algoritmo **CD2S**, lo que significa que el tiempo de ejecución de este algoritmo para la variante 1 es mejor que el tiempo de los demás en cada uno de los casos de prueba utilizados en este experimento. Para la variante 2, en un 6,2 % de las veces es mejor utilizar el algoritmo **IB2S**, que en la Tabla E.20 del Apéndice H.6, corresponde a 10 lotes de la combinación 12 – 1. Esta diferencia con respecto a la variante 1, se debe a la inclusión de una dimensión no espacial ya que para el algoritmo **CD2S** implica un aumento en la región de búsqueda, por lo que se reduce la ventaja de utilizar **CD2S**.

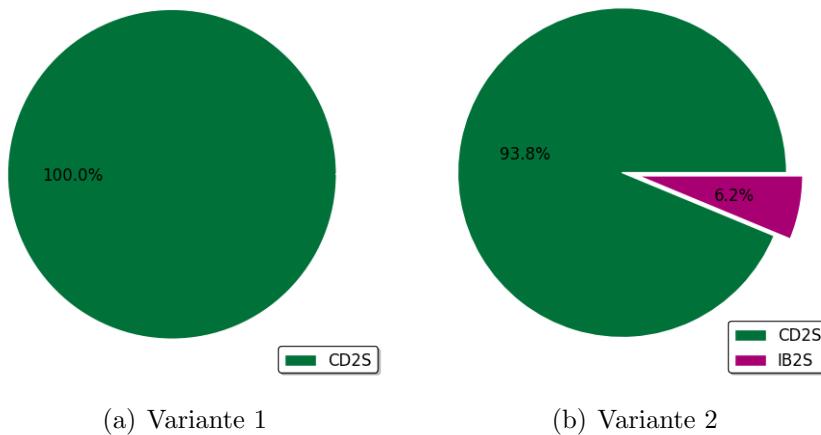
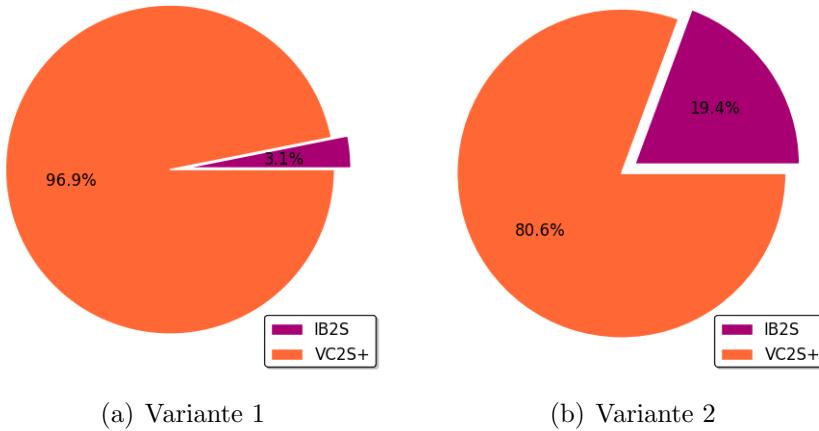


Figura H.23: Porcentaje de lotes en que un algoritmo es el mejor, `vecesTMin`

Contrariamente, en las Figuras H.24(a) y H.24(b), se observa que `vecesTmax` es igual al 96,9 % y 80,6 % de los lotes para el algoritmo **VC2S+** y 3,1 % y 19,4 % para **IB2S**. De esto se infiere que el algoritmo **IB2S** tiene un comportamiento inestable, debido a que el algoritmo **IB2S** mejoró en presencia de dimensiones espaciales, pero también empeoró.

H.6.2 Objetos comparados

Las Figuras H.25(a) y H.25(b), indican el número de objetos que tuvieron que ser comparados para obtener el *skyline*, donde en el eje *x* se muestra la combinación y en el eje *y*

Figura H.24: Porcentaje de lotes en que un algoritmo es el peor, `vecesTMax`

el `#objetos comparados`. Se observa además que para ambas variantes, las líneas de los algoritmos `VC2S+` y `IB2S` están muy cercanas, al contrario de la Figura 5.15 donde están muy alejadas.

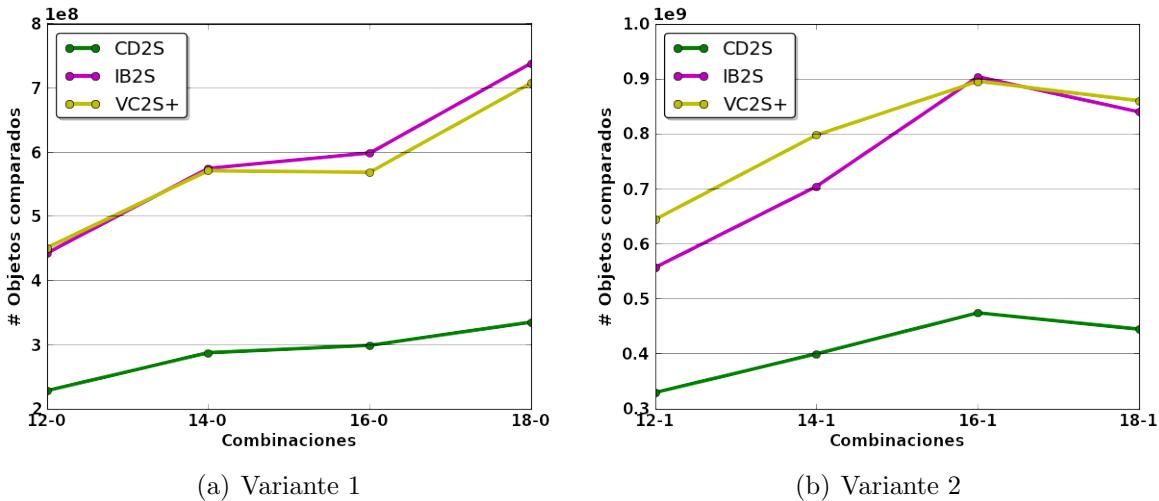


Figura H.25: Objetos comparados en promedio

Este es el motivo por el cual el algoritmo `VC2S+` tiene un peor tiempo de ejecución, ya que aunque realiza menos comparaciones de dominancia que `IB2S`, necesita comparar una cantidad similar de objetos. El algoritmo `VC2S+` en el *modo actualización de estatus*, que es donde obtuvo los peores tiempos, utiliza el algoritmo `VS2` que visita a una mayor cantidad de objetos a diferencia de si lo hiciera con `VCS2` para el *modo actualización de desplazamiento* (en los patrones del 1-5). Por lo tanto, debe navegar más veces en el grafo

Delaunay para obtener los candidatos y comparar los objetos visitados con la ventana del *skyline*, resultando para IB2S menos costoso en tiempo, iterar la lista de candidatos completa y compararlos con la ventana del *skyline*.

H.6.3 Estudio de memoria

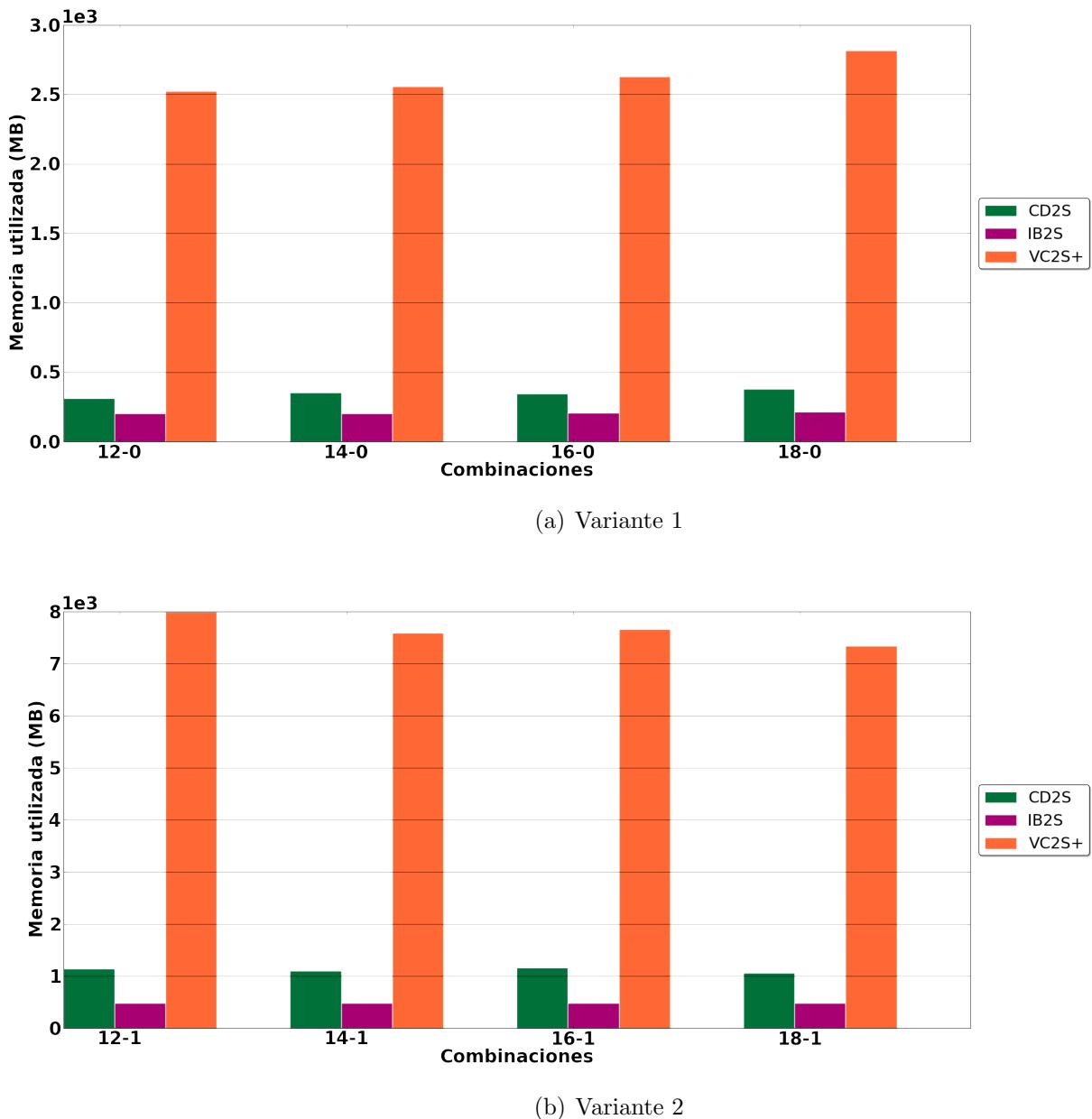


Figura H.26: Memoria promedio por lote

Adicionalmente al análisis realizado en el Experimento VI, se contabilizó la memoria requerida por cada algoritmo para obtener el *skyline* en cada uno de los lotes de 50 simulaciones

de los diferentes casos de prueba para ambas variantes de ese experimento.

En la Figura H.26 se muestra la gráfica de memoria utilizada en promedio en un lote para cada algoritmo y combinación, donde el eje x indica la **combinación** y el eje y la cantidad de **memoria utilizada** en MB .

Se observa que el menor uso de memoria en cada una de las combinaciones, lo tiene el algoritmo **IB2S** debido a que con un procedimiento más básico y menos estructuras obtiene el *skyline*. Para la variante 1 en la Figura H.26(a), el valor es aproximadamente 200 MB y para la variante 2 en la Figura H.26(b), es aproximadamente 480 MB .

Los algoritmos **CD2S** y **VC2S+** necesitan de una mayor cantidad de memoria, 350 MB y 2.600 MB para la variante 1 y 1.110 MB y 7.600 MB para la variante 2, aproximadamente. El algoritmo **CD2S** para mantener todos los índices, las regiones de búsqueda y polígonos de *convex hull* que ayudan a actualizar el *skyline* y el algoritmo **VC2S+** principalmente porque almacena el grafo *Delaunay*, tiene precalculada la lista de vecinos de *Voronoi* para acceder con mayor rapidez a estos objetos y la lista de apuntadores de la celda de *Voronoi* de cada objeto con su distancia mínima a los puntos de referencia.

Se evidencia un incremento al aumentar la cantidad de puntos de referencia, debido a que por cada punto de referencia que sea vértice del *convex hull* se tiene una dimensión a precacular para cada uno de los objetos candidatos. Este incremento es más acentuado para algoritmo **VC2S+** porque además tiene que realizar ese procedimiento en la obtención de las distancias mínimas de cada celda de *Voronoi* a los puntos de referencia. Al mismo tiempo, se observa que entre las variantes existe una gran diferencia en el uso de memoria para los algoritmos en todas las combinaciones, esto se debe a que aumenta la cantidad de candidatos al *skyline* porque la región de búsqueda se construye con los objetos del *skyline* no espacial que pueden estar en cualquier lugar del plano y tiene que ser lo suficientemente grande para incluir los objetos que tienen una buena combinación espacial y no espacial.

H.6.4 Actividad en el *skyline*

Los cambios que realizan los algoritmos en el *skyline*, pueden ser vistos en la Figura H.27, donde la menor cantidad de cambios la realiza el algoritmo CD2S.

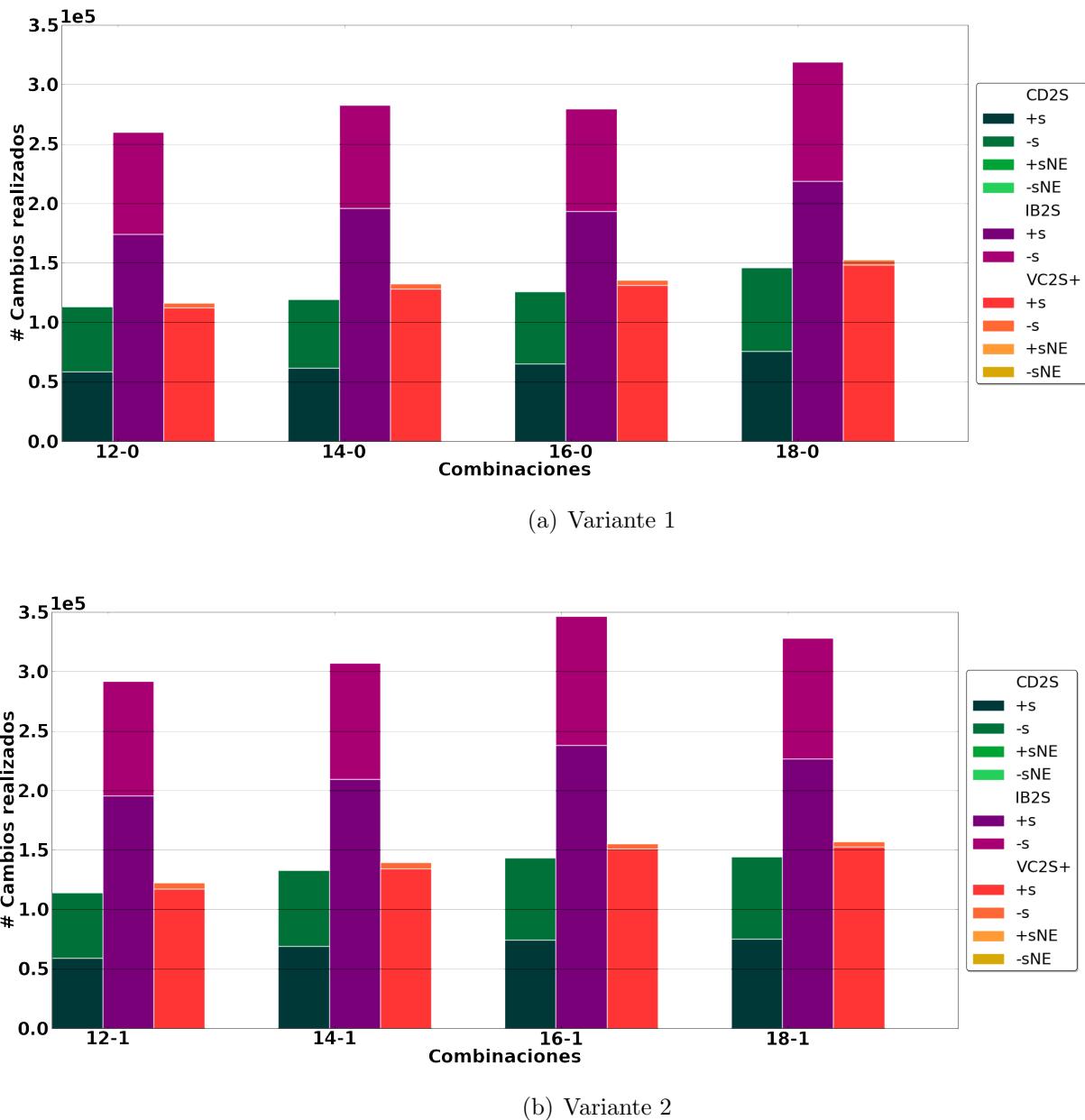


Figura H.27: Cambios en el *skyline* en promedio por lote