Un agente inteligente para la optimización en línea del proceso de mezclas de gasolina

César Pernalete

Facultad de Ingeniería, Centro de Simulación y Modelos,

Universidad de Los Andes

Trabajo presentado como requisito parcial para obtener el grado de

Magister Scientiarum

Mérida 2012

A Ivany, mi esposa

Agradecimientos

Ante todo quiero agradecer a Dios, en principio por haberme dado el gran regalo de la vida y así tener la oportunidad de navegar en el inmenso océano de la ciencia, además, por haberme hecho conocer a Ivany, mi esposa, de quién recibí todo el cariño, apoyo y comprensión que necesité para alcanzar esta meta.

También quiero agradecer a mis padres y a los padres de mi esposa, por haber estado siempre muy atentos y diligentes a prestarme desinteresada ayuda.

Es importante destacar la excelente labor del profesor Jacinto Dávila, mi tutor, de quien siempre recibí oportuna y pertinente asesoría. En la parte académica también recibí un importante apoyo de Dirk Eddelbuetel y Sebástien Le Digabel, de la comunidad de software libre.

Por último quiero agradecer a PDVSA Intevep por haberme brindado la oportunidad y las condiciones necesarias para llevar a cabo este trabajo. Especialmente quiero agradecer a Antonio y a Jesús, sus recomendaciones y enseñanzas fueron definitivamente cruciales.

Resumen

El mezclado de gasolinas es un proceso muy importante en la industria de refinación de crudo. Grandes beneficios económicos como eliminación de los regalos de calidad, eliminación de re-mezclas e incremento de la flexibilidad operacional en el almacenaje son obtenidos al implementar sistemas para la optimización en línea de este proceso. Normalmente estos sistemas utilizan simplificaciones lineales con ajustes no-lineales para modelar el proceso y algoritmos de programación lineal o no-lineal clásicos. Este tipo de estrategias puede generar errores considerables en los resultados del ejercicio de optimización dada las diferencias modelo-proceso. Con el objeto de evitar este inconveniente, en este trabajo se plantea el diseño y desarrollo de un optimizador en línea de mezclas de gasolina, que cuente con modelos que representen de una manera más precisa las propiedades resultantes en una mezcla de combustibles. Para ello se desarrollaron modelos de mezcla de propiedades con máquinas de soporte vectorial (MSV) a partir de datos reales de un proceso real. Para operar las MSV como restricciones de un problema de optimización sobre un espacio de búsqueda de recetas, se plantea el uso del software NOMAD, el cual utiliza el algoritmo de búsqueda directa MADS *(Mesh Adaptative Direct Search)* para realizar la optimización. Tanto el NOMAD, como las MSV se integran en una arquitectura para optimizar el proceso de mezclado, cuyo modelo general es parte de esta propuesta. En ese contexto, el subsistema optimizador es concebido como un agente inteligente que percibe condiciones en su entorno y propone soluciones oportunas. El agente optimizador ha sido implementado y para evaluar su desempeño se han realizado simulaciones de todo el sistema de mezclas en condiciones estáticas y dinámicas bien definidas que incluyen escenarios críticos en el comportamiento del proceso en tiempo real.

Índice general

Lista de figuras

Lista de tablas

Capítulo 1

Introducción

El mezclado de gasolinas es un proceso semi-continuo que consiste en la combinación de un conjunto de productos intermedios generados en el proceso de refinación de petróleo denominados "básicos", con el objeto de crear un producto final con ciertas especificaciones de calidad (Wen & Morales, 2004; Singh *et al.* , 1999; Wang *et al.* , 2007). Un esquema de procesos simplificado de una refinería puede ser observado en la Figura 1.1, donde las líneas oscuras representan los básicos mezclados para la producción de gasolinas terminadas.

Entre las especificaciones de calidad mas importantes actualmente establecidas para gasolinas terminadas se tienen las siguientes propiedades: RVP (Reid Vapor Pressure), IAD (Anti detonant index), RON (Research Octane Number) y destilación D-86. Es importante destacar que cada uno de estos básicos posee propiedades y costos diferentes, por lo que para ser capaz de alcanzar un producto terminado con las especificaciones requeridas, al menor costo posible y con los básicos disponibles en la refinería, es necesario resolver un problema de optimización (Singh *et al.* , 1999; Barsamian, 2003; Wang *et al.* , 2007).

Existen diversas gasolinas terminadas que pueden ser producidas en una refinería dependiendo del requerimiento del cliente. Cada una de estas gasolinas terminadas atiende restricciones de calidad particulares. El cálculo de las proporciones de los básicos bajo el cual se realizará una mezcla para la obtención de una gasolina terminada en específico se ejecuta en un proceso de trabajo denominado programación ó *"scheduling"*. En este proceso se calcula las proporciones bajo las cuales se realizará cada mezcla durante un período que oscila entre un día y una semana (Barsamian, 2003). A las proporciones volumétricas de los básicos en cada una de las mezclas se les denomina "recetas".

Tal como se mencionó anteriormente, la programación consiste en gran medida de la resolución de un problema de optimización fuera de línea (Singh *et al.* , 1999), don-

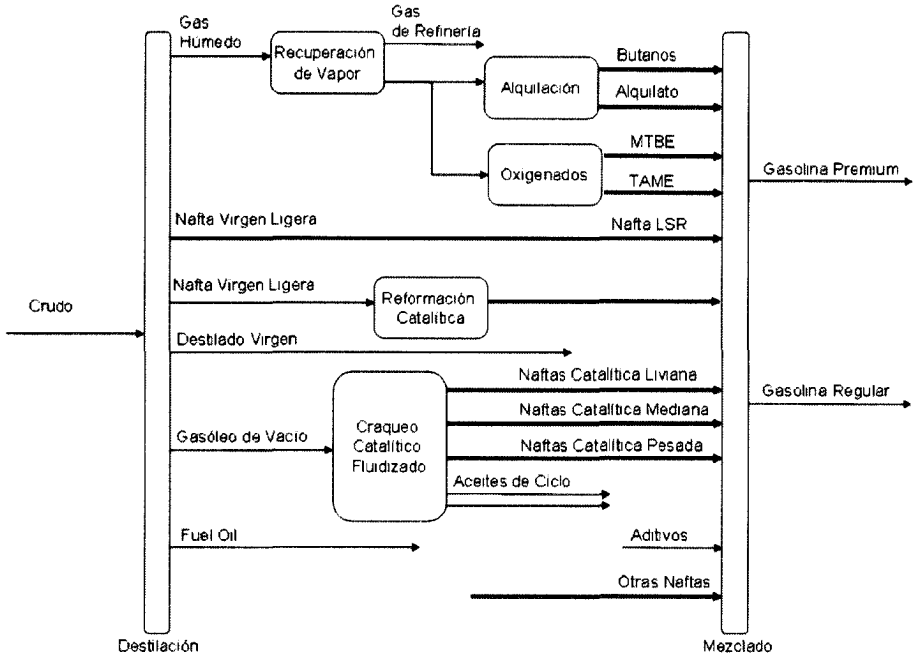

Figura 1.1: Esquema general de una refinería de petróleo

de los modelos tradicionalmente utilizados en la industria para representar la mezcla de básicos y predecir las propiedades del producto final suelen ser puramente lineales o bien lineales para algunas variables con ajustes no-lineales para otras (Singh *et al.* , 1999; Wang *et al.* , 2007). El error de modelado implícito en estas suposiciones frecuentemente conlleva a la obtención de una mezcla fuera de las especificaciones esperadas, generando altos costos adicionales debido a retrasos en la producción causados al corregir la mezcla antes de su colocación en el mercado, o bien por la venta de un producto con "regalos" de calidad, es decir, un producto con una calidad mayor a la esperada por el cliente (Vermeer *et al.* , 1997).

Este problema ha sido abordado por otros investigadores como uno de optimización en tiempo real (OTR), o bien optimización de operaciones en línea (Forbes *et al.* , 2002; Diaz & Barsamian, 1996; Monder, 2001; Wang *et al.* , 2007; Singh *et al.* , 1999; Sullivan, 1990). La OTR es un enfoque de control de procesos basado en modelos, que utiliza información actual del proceso (ej. datos económicos y un modelo del proceso) para predecir las políticas de operación óptima para una unidad de proceso durante el próximo intervalo de OTR (Forbes *et al.* , 2002). En este sentido la optimización en tiempo real del proceso de mezclas representa una oportunidad clave para la minimización de costos en el proceso de refinación, específicamente en el proceso de mezcla

de combustibles para efectos de esta tesis.

Si bien la optimización en tiempo real ha sido una estrategia de provecho para la solución del problema referente a la obtención de mezclas fuera de especificación, la importancia de un modelo "adecuado" para obtener un buen desempeño de la estrategia de OTR es bien conocida (Forbes & Marlin, 1994), y generalmente los modelos utilizados en estas aplicaciones son los mismo empleados en el caso de la optimización fuera de línea lo cual podría generar un desempeño no satisfactorio de la estrategia de optimización. En tal sentido en este trabajo se propone desarrollar modelos que representen de una manera más precisa la mezcla de combustibles, con el objeto de tener estimaciones más adecuadas de las propiedades de la mezcla final, e incrementar así la eficacia de las estrategia de OTR.

1.1. El proceso de mezclas de gasolina

Un esquema del proceso de mezcla de gasolinas estándar puede ser apreciado en la Figura 1.2, donde se muestra un conjunto de corrientes de alimentación C_1, C_2, ..., C_n que convergen en un cabezal de mezcla común C_m. Cada una de estas corrientes proviene de diferentes etapas del proceso de refinación tal como destilación, craqueo catalítico, reformación catalítica, hidrocraqueo y alquilación, entre otros. Cada uno de estos flujos de alimentación representa los básicos mencionados previamente y posee un conjunto de propiedades (o calidades) asociadas. Estas propieades se denotan con la letra P_{ij} en la Figura 1.2, donde $i \in [1, n]$ identifica la corriente correspondiente y $j \in [1, k]$ la propiedad particular. Existe una gran cantidad de propiedades utilizadas para especificar la calidad de una gasolina, tal como el contenido de olefinas, aromáticos y benceno; sin embargo, las más importantes debido al coste económico para cumplir con su especificación en el producto final son RON, MON, IAD, RVP y destilación D-86. Asimismo, estas propiedades describen un comportamiento claramente no lineal en la mezcla lo cual las hace difíciles de modelar y por lo tanto representan un problema especial cuando estos modelos deben ser utilizados como parte de un ejercicio de optimización. Seguidamente se describe brevemente cada una de ellas.

El número de octano es una medida de la capacidad antidetonante de un combustible. Dos procedimientos de prueba estándar son utilizados para caracterizar la capacidad antidetonante de un combustible para motores de ignición por chispa: la prueba ASTM-908 a través del cual se obtiene el Research Octane Number (RON) y la prueba ASTM-357 a través de la cual se obtiene el Motor Octane Number (MON).

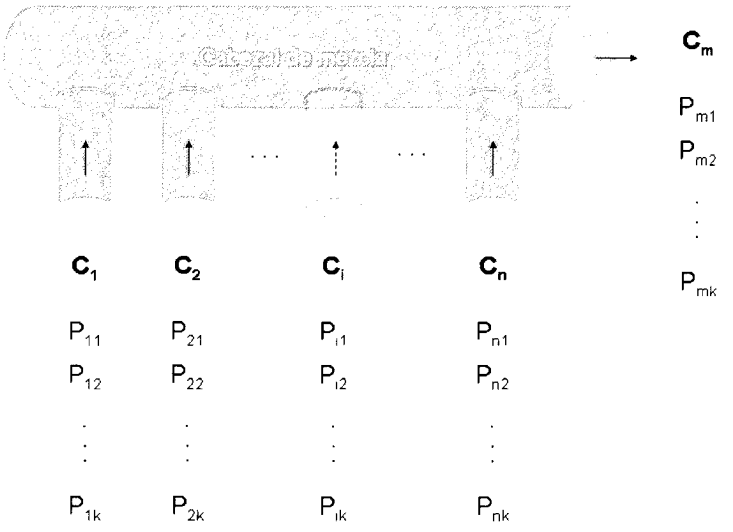

Figura 1.2: Proceso de mezcla de gasolina

Asimismo el IAD representa el promedio aritmético de estas dos propiedades (Singh *et al.* , 1999; Gary & Handwerk, 2001). Otras propiedades importantes que afectan el desempeño de un motor son la volatilidad y el rango de ebullición. La presión de vapor Reid (RVP) es un indicador de la volatilidad de una mezcla de gasolina y es aproximadamente la presión del vapor a 38^{o}C (Wen & Morales, 2004). Por otra parte, las gasolinas, por ser una mezcla de hidrocarburos, mas allá de poseer un único punto de ebullición poseen un rango de ebullición el cual es un indicador de los componentes presentes en una mezcla. Esta propiedad es caracterizada con el procedimiento estándar ASTM-D86 (Gary & Handwerk, 2001).

1.2. Modelos de propiedades de mezcla de combustibles

La predicción de las propiedades de las mezclas no es una tarea trivial, ya que la mezcla no es lineal, es decir, la propiedad de la mezcla resultante no es un promedio ponderado con respecto de las propiedades y las proporciones de los componentes de la mezcla. Para manejar esta no-linealidad normalmente se utilizan correlaciones de propiedades no-lineales, siendo las más populares el conjunto de ecuaciones Ethyl

R-70 para octanaje, y el método de interacción de DuPont para número de octano, presión de vapor Reid y destilación (Barsamian, 2003). También existen estudios donde se han propuesto modelos basados en máquinas de aprendizaje como las redes neuronales artificiales, con el objeto de predecir propiedades de mezclas de gasolinas (Wen & Morales, 2004).

A continuación se describen brevemente las correlaciones normalmente utilizadas para la predicción de propiedades de mezcla de gasolinas.

1.2.1. Número de octano

El número de octano es una medida de la capacidad antidetonante de un combustible, es decir, su capacidad para resistir la detonación durante la combustión en la cámara de combustión (Singh, 1997). Tal como se mencionó anteriormente, existen dos procedimientos de prueba estándar utilizados para caracterizar la capacidad antidetonante de un combustible para motores de ignición por chispa: la prueba ASTM-908 a través del cual se obtiene el *Research Octane Number* (RON) y la prueba ASTM-357 a través de la cual se obtiene el *Motor Octane Nuber* (MON) (Gary & Handwerk, 2001). Los modelos principalmente utilizados para predecir el número de octano en mezclas se describen a seguidamente.

1.2.1.1. Ethyl R-70

El método Ethyl es uno de los más antiguos disponibles en la literatura. Este método ha sido utilizado como una referencia contra la cual nuevos modelos han sido comparados. El octanaje de la mezcla es modelado explícitamente como una función de la sensibilidad de los componentes (la diferencia entre RON y MON), el contenido de olefinas y el contenido de aromáticos (Wen & Morales, 2004).

$$RON_{blend} = \bar{r} + a_1(\overline{rs} - \bar{r}\bar{s}) + a_2((\overline{O^2}) - (\bar{O})^2) + a_3((\overline{A^2}) - (\bar{A})^2) \tag{1.1}$$

$$MON_{blend} = \bar{m} + a_4(\overline{ms} - \bar{m}\bar{s}) + a_5((\overline{O^2}) - (\bar{O})^2)$$

$$+a_6\left(\frac{(\overline{A^2}) - (\bar{A})^2}{100}\right)^2 \tag{1.2}$$

En la expresión anterior, r es *RON*, m es *MON*, s es la sensibilidad *(RON - MON)*, O es contenido de olefinas (% Volumen), A es contenido de aromáticos (% Volumen),

y a_1, a_2, a_3, a_4, a_5, a_6 son coeficientes de correlación. Todas las variables con barras superiores representan promedios volumétricos (Monder, 2001).

Considerando mezclas de gasolina sin plomo similares a aquellas utilizadas para generar los coeficientes de correlación, el error estándar de predicción de estos modelos se reporta en 0,66 y 0,85 numero de octano para RON y MON respectivamente. Para otras mezclas el error estándar de predicción se encontró en 0,92 para RON y 0,61 para MON (Singh, 1997).

1.2.1.2. Modelo de interacción

Este modelo se basa en modelos de dos factores donde el efecto global se atribuye al efecto principal de cada uno de los dos factores y la no linealidad se captura como un término de interacción. El término de interacción considera el efecto de un factor sobre el otro y puede ser determinado utilizando un diseño experimental apropiado. Por lo tanto, en el método de interacción, la no-linealidad en el octano de una mezcla de gasolina respecto de las composiciones de los básicos se atribuye a las interacciones de dos factores entre los componentes de las mezcla. Esta no-linealidad se considera añadiendo un término de interacción al promedio volumétrico del número de octano (Singh, 1997). Para un sistema de n básicos de mezcla, el número de octano (RON ó MON) de la mezcla se puede calcular de la siguiente forma:

$$ON = \sum_{i=1}^{n} x_i p_i + \sum_{k=i+1}^{n} I_{i,k} x_i x_k \tag{1.3}$$

En la expresión anterior, ON es el número de octano de la mezcla, p es el número de octano de cada básico, x es la proporción volumétrica de cada básico, $I_{i,k}$ es el coeficiente de interacción entre dos componente i y k, el cual es dado como:

$$I_{i,k} = 4O_{i,k} - 2(O_i + O_k) \tag{1.4}$$

donde $O_{i,k}$ es el número de octano de una mezcla en partes iguales $(50:50)$ entre los componentes i y k, mientras que O_i y O_k es el número de octano para el componente i y k respectivamente (Singh, 1997; Wen & Morales, 2004).

En un estudio realizado por Morris en 1975, se estudia la exactitud de los métodos de interacción y se comparan con el enfoque del método Ethyl RT-70. Se reporta en este estudio que a pesar de que los modelos de interacción se ajustan de una mejor manera a los datos experimentales en comparación con el método Ethyl RT-70. La exactitud de la predicción para datos generales no utilizados en la calibración del

modelo es muy similar en ambos casos. El error estándar de predicción reportado en este estudio se encuentra entre 0.6 y 1.0 número de octano (Singh, 1997).

1.2.2. RVP

La presión de vapor Reid (RVP, por sus siglas en inglés), definida por la Sociedad Americana para la Evaluación de Materiales bajo la designación ASTM D-323-56, es un indicador de la volatilidad de una mezcla de gasolina y es aproximadamente la presión del vapor a 38^{o}C (100^{o}F) (Singh, 1997; Wen & Morales, 2004). El RVP de una gasolina afecta su desempeño en los motores de combustión en términos arranque, calentamiento y velocidad de aceleración (Gary & Handwerk, 2001). El RVP es importante desde el punto de vista económico ya que su especificación máxima de RVP limita la cantidad de n-butano añadida a mezcla, el cual se muestra como una fuente económica de octano (Singh, 1997).

Los modelos principalmente utilizados para modelar la presión de vapor Reid (RVP) se describen a continuación.

1.2.2.1. Método del índice de mezcla

El método de índice de mezcla es un método empírico fácil de utilizar desarrollado por *Chevron Research Company* (Gary & Handwerk, 2001). Este modelo es ampliamente utilizado en el ámbito industrial por su simplicidad algebraica (Wen & Morales, 2004). En este enfoque los RVP de las mezclas se predicen utilizando los índices de mezcla de presión de vapor Reid (RVPBI, por sus siglas en inglés) los cuales mezclan linealmente. El cálculo de los RVPBI se describe con detalles en (Gary & Handwerk, 2001), sin embargo, comunmente son de la siguiente forma:

$$RVPBI_i = p_i^{1,25} \tag{1.5}$$

y el cálculo del RVP de la mezcla se completa de la siguiente forma:

$$(RVP)_{mezcla} = \sum_{i=1}^{n} (x_i \cdot RVPBI_i) \tag{1.6}$$

En el trabajo realizado por (Wen & Morales, 2004) se muestra una pequeña variación del método del índice de mezcla al añadirle un exponente global. Una expresión típica de este método sería la siguiente:

$$RVP = \left(\sum_{i=1}^{n} (x_i p_{bi}) \right)^{0.8} \tag{1.7}$$

En esta expresión el exponente de la expresión global es ajustado de acuerdo a la experiencia particular en cada aplicación. En este caso $p_{bi} = p_i^{1,25}$ es el RVPBI.

1.2.2.2. Modelo de interacción

Según (Singh, 1997) en un estudio realizado por Morris en 1975 el enfoque de interacción planteado para número de octano, es también válido para predecir RVP en mezclas. Por lo tanto, de forma similar al caso del número de octano, para el RVP el modelo de interacción se plantea como sigue:

$$RVP = \sum_{i=1}^{n} x_i P_i^v + \sum_{k=i+1}^{n} I_{i,k} x_i x_k \tag{1.8}$$

donde $I_{i,k}$es el coeficiente de interacción entre el componente i y k, el cual es dado como:

$$I_{i,k} = 4P_{i,k}^v - 2(P_i^v + P_k^v) \tag{1.9}$$

En la expresión anterior $P_{i,k}^v$ es la presión de vapor Reid de una mezcla 50 : 50 de los componentes i y k, mientras que P_i^v y P_k^v son la presión de vapor Reid para el componente i y k respectivamente (Singh, 1997; Wen & Morales, 2004).

En el mismo estudio de Morris se indica que el error estándar de predicción de este método en un análisis de 42 mezclas de 8 componentes es 0,18 psi (Singh, 1997).

1.2.3. Destilación

Las gasolinas, por ser una mezcla de hidrocarburos con diferentes puntos de ebullición, mas allá de poseer un único punto de ebullición, tal como ocurre para los compuestos puros, poseen un rango de ebullición el cual es un indicador de los compuestos presentes en una mezcla (Rubio, 2004). Un rango destilación típico de una gasolina comercial se encuentra entre 30^{o}C para el punto inicial de ebullición (PIE) y 200^{o}C para el punto final de ebullición (PFE) (Rubio, 2004), mientras que para una nafta liviana el rango de ebullición se encuentra entra un PIE de 30^{o}C y un PFE de 90^{o}C (Singh, 1997).

La volatilidad característica de una gasolina es medida a través del procedimiento estándar de destilación ASTM-D86 el cual es una destilación por lotes o destilación *batch* llevada a cabo en el laboratorio a presión atmosférica y sin fraccionamiento (Singh, 1997), es decir, sin separar los componentes en diversos sub-productos de acuerdo a algún rango de ebullición. Esta propiedad además de indicar el PIE y el

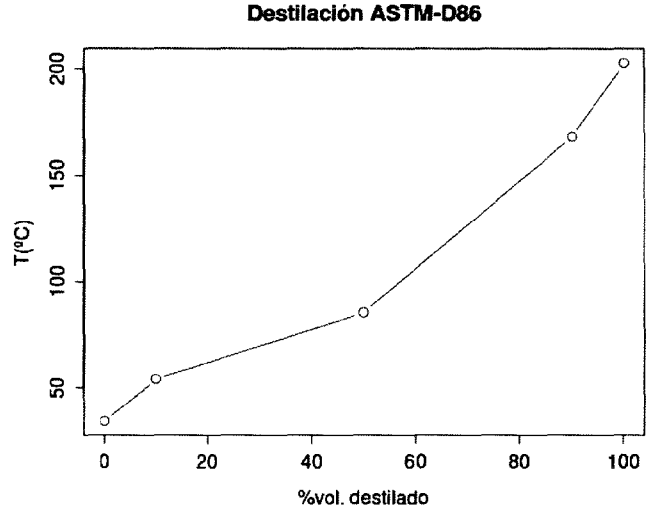

Figura 1.3: Destilación ASTM típica para una gasolina terminada

PFE, describe puntos intermedios de la destilación que corresponden a la temperatura para un porcentaje de volumen destilado dado (Rubio, 2004). Los puntos normalmente especificados como resultados del ensayo D-86 son PIE, T10 %, T50 %, T90 % y PFE, donde TXX % representa la temperatura para un XX % de volumen destilado. El gráfico %volumen destilado vs. temperatura normalmente luce una tendencia sigmoidal y un caso típico para una gasolina terminada se muestra en la Figura 1.3.

Con respecto al cálculo de destilación de mezclas, el método de interacción para la destilación se podría aplicar de forma similar al caso del número de octano y presión de vapor Reid. Sin embargo, existe una gran desventaja en la aplicación de este método que radica en la gran cantidad de parámetros de interacción a determinar, lo que implica un gran esfuerzo de trabajo en el laboratorio (Singh, 1997). Por otra parte se debe tener en consideración que no ofrece resultados muy confiables para los puntos iniciales y finales (PIE y PFE) de la destilación por lo que sólo puede ser utilizado para puntos intermedios (Rubio, 2004).

1.3. Optimización en tiempo real de mezclas

1.3.1. Aspectos de interés en el proceso de mezcla

El mezclado de gasolinas es un proceso que puede ser considerado como un proceso por lotes, donde la producción (volumen mezclado) se fija ya sea contractualmente, o por el cronograma de producción de la refinería. Cualquier mezcla debe alcanzar

ciertos parámetros de calidad, los cuales son dependientes del grado de la gasolina. Además, este proceso está sujeto a un número de restricciones operacionales que incluye la disponibilidad de los básicos o componentes de la mezcla y las facilidades de almacenamiento de producto (Singh *et al.* , 1999).

El desempeño de un mezclador de gasolinas, o un sistema automatizado de mezclas, es frecuentemente discutido en términos de "regalos de calidad" del producto y el número de re-mezclas requeridas por mes. Los "regalos de calidad" se refieren a preparar un producto con calidades que exceden el mínimo requerido (o exceden el máximo requerido dependiendo de la variable en consideración) (Wang *et al.* , 2007; Singh *et al.* , 1999; Vermeer *et al.* , 1997). Tal "regalo de calidad" puede indicar que cierta cantidad de un componente más costoso fue utilizado cuando un componente de menor costo pudo haber sido utilizado en su lugar, posiblemente resultando en una disminución de la rentabilidad de la mezcla (Singh *et al.* , 1999).

1.3.2. Estrategias de optimización en tiempo real de mezclas

Existen diversas alternativas de soluciones comerciales ofrecidas para automatización de mezclas. En (Anonimo, 1997) se puede encontrar una descripción detallada de las soluciones principalmente usadas en la industria de refinación. Asimismo, la arquitectura de optimización y control de mezclas clásica encontrada en las refinerías se muestra en la Figura 1.4. Allí se puede observar que la arquitectura se plantea en tres niveles: un primer nivel de programación, un segundo nivel de optimización y un tercer nivel de control regulatorio (Sullivan, 1990; Singh *et al.* , 1999).

El primer nivel, denominado programación, consiste en la ejecución de una optimización fuera de línea *(off-line)* donde se planifica las operaciones de mezclado sobre un período de tiempo menor a una semana (Singh *et al.* , 1999; Barsamian, 2003). En esta etapa se asume que las propiedades de los básicos poseen un valor fijo, el cual normalmente se considera invariable en un período de tiempo no mayor a un mes. Con esta información, e información adicional como inventarios y costos de básicos, se calculan las recetas iniciales que se descargan al optimizador en línea (Singh *et al.* , 1999; Barsamian, 2003).

El segundo nivel, denominado optimización en línea, utiliza información en línea correspondiente a las calidades de la mezcla, obtenidas a través de un analizador en línea de propiedades, para modificar ya sea la receta inicial o las recetas que deberá ejecutar el controlador en el transcurso del proceso. Los optimizadores en línea están concebidos ya sea para minimizar la desviación frente a la receta inicial o para optimizar algún indicador de desempeño económico (como rentabilidad de la

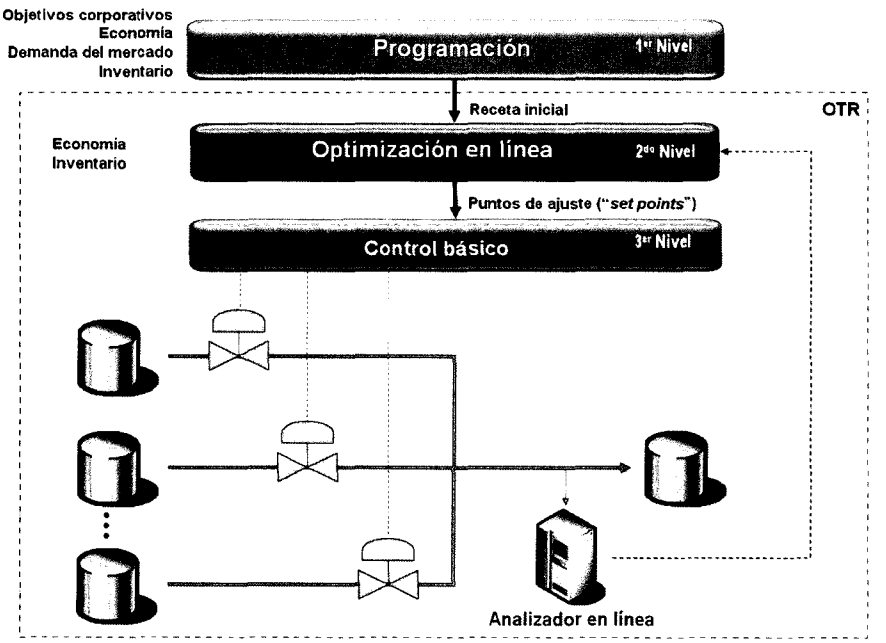

Figura 1.4: Jerarquía de control de mezclas
Fuente: Singh (1997)

mezcla) mientras se satisfacen las restricciones de calidad del producto (Michalek *et al.* , 1994).

El tercer nivel está conformado por sistemas de control distribuido (DCS, por sus siglas en inglés) basado en microprocesadores y otros sistemas de instrumentación convencionales (Sullivan, 1990).

Los récipes de mezcla generados por el optimizador en línea proveen los puntos de ajuste de los flujos de los componentes en la mezcla, los cuales son implementados a nivel de control regulatorio a través del DCS. Los diferentes sub-sistemas de control y optimización deben estar integrados apropiadamente para proveer la información requerida correspondiente al proceso, necesaria para asegurar el éxito de todo el sistema. Bases de datos son utilizadas para mantener un seguimiento de los inventarios de básicos, flujos de corrientes y calidades del producto mezclado. Cada subsistema obtiene u ofrece información de las bases de datos vinculándose cada subsistema hasta crear una red integrada (Singh *et al.* , 1999). Tal como se puede observar en la Figura 1.3, el corazón del sistema de automatización de mezclas es el sistema de optimización en tiempo real (OTR), el cual está conformado por el optimizador en línea y el sistema de control regulatorio (Singh *et al.* , 1999).

En la Figura 1.4 se puede observar un ejemplo típico de un reporte final de una

MEZCLADOR DE GASOLINAS - REPORTE FINAL DE MEZCLA

Número de mezcla	88008	Fecha de inicio	50062010
Código del producto	23605/1	Hora inicial	08:30
Volumen del lote (BBL)	80000	Fecha de culminación	50062010
Tanque de producto	T-48	Hora Final	16:35

TANQUE FINAL			Especificación			TANQUE FINAL		Regalo
Componente	Volumen (BBL)	Receta (%v/v)	del producto	Min	Max	Predicción	Laboratorio	de calidad
Nafta Liviana Virgen	17200	21,5	RON	92		92,4	92,3	0,3
Nafta Catalítica	34800	43,5	MON	80		80,3	80,2	0,2
Reformado	12160	15,2	RVP	6,97	9,15	9,15	8,91	
MTBE	4640	5,8	IBP (°C)		35		33	
TAME	9360	11,7	T10 (°C)		70	52	54	
Butano	1840	2,3	T50 (°C)		140	99	87	
Alquilato	0	0,0	T90 (°C)		195	158	170	
			PFE (°C)		210	208	198	
TOTAL	80000	100	Densidad (g/cc)	0,735	0,78	0,75	0,75	

Figura 1.5: Reporte final típico de una mezcla de gasolinas automatizada

mezcla automatizada. Allí se muestran todos los datos de interés de un proceso de mezcla tal como volumen mezclado, receta final de la mezcla, calidad del producto real y predicha, especificaciones del producto y regalos de calidad. Nótese que una receta no es más que la proporción volumétrica de los componentes que conforman una mezcla.

Todos los modelos de optimización en línea actualmente utilizados, se basan en programas matemáticos con recetas X como las variables de decisión. Un modelo de programación lineal es el más simple utilizado en sistemas de mezclas de gasolina, donde tanto la función objetivo como las restricciones son lineales (Wang *et al.* , 2007). En aplicaciones reales, una actualización de ajuste o sesgo (*bias*) se agrega a las restricciones para disminuir las diferencias modelo-proceso (Forbes & Marlin, 1994; Wang *et al.* , 2007), y la formulación esta dada por la ecuación (1.10).

$$\begin{aligned} &\max_{x} (CX) \\ &QX \leqq (e^t X)S + \beta \\ &HX \leqq \rho \end{aligned} \tag{1.10}$$

En la ecuación (1.10) C representa el vector que contiene los precios tanto para los productos terminados como para la corriente de alimentación, X es el vector de flujos de las corrientes de alimentación, Q es una matriz que contiene las propiedades de las corrientes de alimentación, S es la matriz de especificaciones de calidad, β es

el vector de términos de ajuste o *bias* para ajustar los modelos de mezcla, H es la matriz de restricciones de demanda de productos y disponibilidad de insumos, ρ es el vector de demanda de productos y disponibilidad de insumos y e es un vector de pesos (normalmente contiene sólo unos y ceros para indicar la presencia de una corriente específica en la mezcla). La letra t únicamente indica una operación de trasposición del vector e.

Cuando los modelos de optimización utilizan los modelos de mezcla no-lineales indicados en la sección 1.2, la diferencia con respecto a la formulación anterior es que QX es reemplazada por una función no-lineal $g(Q, X)$ (Wang *et al.* , 2007). Replanteando la ecuación (1.10) para el caso de modelos de mezcla no-lineal se tiene la ecuación (1.11).

$$\begin{aligned} &\max_{x} (CX) \\ &g(Q, X) \leqq (e^t X)S + \beta \\ &HX \leqq \rho \end{aligned} \tag{1.11}$$

La resolución del problema de optimización mostrado en la ecuación (1.10), se enfoca a través de técnicas convencionales de programación lineal como el método simplex (Barsamian, 2003), mientras que para el problema de optimización mostrado en la ecuación (1.11) normalmente se utilizan aplicaciones comerciales como GAMS (General Algebraic Modeling Systems), MINOS o Lancelot (Singh *et al.* , 1999) que implementan variaciones de métodos de programación lineal como programación lineal sucesiva (SLP, por sus siglas en inglés), programación cuadrática sucesiva (SQP, por sus siglas en inglés) ó programación de gradiente reducido generalizado (GRG, por sus siglas en inglés) (Barsamian, 2003).

1.3.3. Visión combinatoria de la complejidad del espacio solución

El cálculo de una receta consiste en hallar una combinación adecuada de básicos, que permita, en principio, obtener una mezcla cuyas propiedades se encuentren dentro de ciertas especificaciones de calidad. Definiendo con más detalles el problema, basándonos en la Figura 1.2, se puede decir que se debe hallar una combinación adecuada de proporciones de i corrientes $C_1, C_2, ..., C_i$, donde cada una de estas corrientes está caracterizada por k propiedades $P_1, P_2, ..., P_k$, y en la mezcla final C_m, estas k propiedades deben cumplir ciertas restricciones de calidad.

Por simplicidad se podría suponer que cada propiedad P_k oscila entre 0 % y 100 % de un rango definido. Así que si se pudiera tene una tabla permanente que relacione las proporciones de las i corrientes de entrada de acuerdo a cada una de sus k propiedades, con cada una de las k propiedades de la corriente de salida, se tendrían k tablas con 100^{2i+1} filas. Si $i = 8$, como es el caso de este estudio, $100^{2i+1} = 100^{17} = 10^{34}$, un número definitivamente muy grande de posibilidades para explorar por cada tabla.

Sin embargo, se podría plantear una simplificación sensata para nuestro caso de estudio. En el planteamiento previo, se describen las corrientes de entrada como si sus propiedades pudiesen oscilar en un rango muy amplio. Si bien las propiedades para una corriente pueden variar, para una refinería en particular sus valores tienden a ser mas o menos constantes, hasta el punto en el que se pueden considerar invariables. Esta suposición es completamente válida para refinerías venezolanas cuya alimentación de crudo es muy constante, lo que resulta en productos intermedios con poca variabilidad en sus propiedades.

En este nuevo caso simplificado sólo se relacionaría las proporciones de las i corrientes de entrada, con cada una de las k propiedades de la corriente de salida, por lo que el planteamiento combinatorio con respecto de la totalidad de posibilidades de mezcla sería $100^i \cdot 100 = 100^{i+1} = 100^{8+1} = 10^{18}$. Si bien este es un número mucho menor al obtenido previamente, de igual manera es un número muy grande. Asumiendo que un computador tarde 1 milisegundo para revisar cada opción, tomaría 10^{18} milisegundos para revisar todas las opciones, que es aproximadamente igual a $3 \cdot 10^7$ años, definitivamente un tiempo inadecuado para nuestro propósito.

En cualquier caso, la reducción de la complejidad del problema basada en la poca variabilidad de las propiedades de las corrientes de entrada, es un aspecto que debe ser tomado en cuenta en este estudio para el modelado de las propiedades de mezcla.

1.4. Objetivos y alcance de esta tesis

En este trabajo de tesis se propone el diseño y desarrollo de la capa responsable de realizar optimización en tiempo real en el proceso de mezclas de gasolina. Tal como se planteó en secciones previas, los modelos utilizados en los optimizadores en línea de mezclas actuales utilizan generalmente modelos lineales, modelos lineales con ajustes no-lineales, o aproximaciones empíricas no-lineales para representar la mezcla de propiedades, siendo esto un problema ya que estas diferencias modelo-proceso conllevan frecuentemente a la generación de productos fuera de especificación con las consecuentes pérdidas económicas. El optimizador propuesto en este trabajo utiliza

aprendizaje estadístico, específicamente la técnica de Máquinas de Soporte Vectorial (MSV), para construir modelos de mezcla de propiedades a partir de información histórica del proceso. A través de esta estrategia se obtuvieron modelos de mezcla mucho más precisos, es decir, con menores diferencias modelo-proceso, y que por lo tanto mejoran los resultados en el cálculo de las recetas que deben ser indicadas al nivel de control en cada uno de los pasos de la optimización.

En el capítulo 2 de este trabajo se realiza una introducción a los aspectos teóricos relacionados con las Máquinas de Soporte Vectorial pertinentes a este trabajo. Las MSV son una técnica de modelado ampliamente utilizada para resolver problemas de clasificación y regresión. Tomando en cuenta que el foco de este trabajo es la regresión a partir de datos históricos para la posterior predicción de propiedades, el fundamento teórico descrito está principalmente enfocado hacia regresión utilizando MSV. Asimismo, en este capítulo se describe la herramienta "R", un *software* libre para estudios estadísticos que fue utilizado en este trabajo para implementar los modelos basados en máquinas de aprendizaje.

En la sección 1.3.2 se mostraron las técnicas de optimización utilizadas cuando las restricciones manejadas son algebraicas lineales o no-lineales. En este caso las restricciones del proceso, si bien son algebraicas, son muy difíciles de manejar analíticamente para obtener las derivadas o segundas derivadas, información esencial para aplicar las algoritmos de optimización no-lineal clásicos, por lo tanto para la resolución de este problema es necesario un algoritmo alternativo. En el capítulo 3 se realiza una descripción del algoritmo utilizado para la resolución del problema de optimización, el algoritmo MADS *(Mesh Adaptive Direct Search)*. De igual forma, se hace una reseña de la aplicación utilizada como implementación de este algoritmo, "NOMAD".

En el capítulo 4 se retoma la discusión del problema de la optimización en línea de mezclas y se describe la estrategia utilizada para implementar el método propuesto, es decir, el agente inteligente. En el capítulo 5 se muestra un caso de estudio donde se realiza la implementación del agente inteligente para la optimización en línea de mezclas. Se muestran detalles de implementación donde se aborda el problema de conectividad entre las herramientas "R" y "NOMAD", y se muestra la efectividad de la estrategia propuesta. En el capítulo 6 se presentan las conclusiones obtenidas de la ejecución de este trabajo tanto de la implementación del agente inteligente como de la efectividad de la estrategia propuesta. También se discute la orientación de futuros trabajos en esta dirección.

Los resultados de este trabajo pueden ser directamente extendidos a otros procesos de mezcla, mas allá de la mezcla de combustibles para producir gasolinas termina-

das. Asimismo, pueden servir como referencia para la implementación de estrategias de optimización en tiempo real de procesos basadas en agentes inteligentes, y para la resolución de problemas de optimización donde las funciones utilizadas como restricciones sean desconocidas o difíciles de manejar analíticamente.

Capítulo 2

Modelos basados en Máquinas de Soporte Vectorial

El modelado matemático exacto de mezclas de gasolina es demasiado complejo para ser manejado analíticamente (Wen & Morales, 2004). Con la finalidad de ejecutar actividades de "control de mezclas basado en modelos", un método comúnmente utilizado es la predicción de las propiedades de mezcla a través de la aproximación con un modelo lineal (Wen & Morales, 2004). Sin embargo, las propiedades más importantes de las gasolinas presentan no-linealidades en la mezcla, por lo que la aproximación previamente descrita no ofrece buenos resultados (Barsamian, 2003).

Con el objeto de obtener modelos de mezcla más precisos, varias propuestas se han visto en la literatura para la creación de modelos de mezcla de propiedades utilizando datos operacionales. En el trabajo realizado por (Wen & Morales, 2004), se propone el modelado de propiedades de mezcla utilizando redes neuronales. De forma similar, en (Rubio, 2004), se propone el modelado de mezcla de propiedades de crudo utilizando la misma técnica. En ambos casos los autores manifiestan haber obtenido buenos resultados, por lo que los modelos basados en aprendizaje automático se muestran como una buena opción para modelar propiedades de mezcla de hidrocarburos.

Es importante destacar que las máquinas de soporte vectorial (MSV) presentadas por Vapnik y colaboradores son un método muy poderoso para realizar clasificación y regresión a partir de datos experimentales. En pocos años desde su introducción ya ha superado la mayoría de los sistemas de aprendizaje en un amplio campo de aplicaciones (Cristianini & Shawe-Taylor, 2000). Es por ello que en este trabajo se propone la utilización de las máquinas de soporte vectorial para el modelado de propiedades de mezclas de gasolina. En tal sentido, en este capítulo se ofrece una introducción a la fundamentación teórica que soporta esta técnica.

2.1. Máquinas de soporte vectorial para regresión

El algoritmo de soporte vectorial es una generalización para el caso no-lineal del "algoritmo marco generalizado" desarrollado en Rusia en los sesenta por Vapnik y Chervonenkis. Como tal está firmemente soportado en la teoría de aprendizaje estadístico o teoría VC la cual ha estado siendo desarrollada durante las últimas tres décadas por Vapnik, Chervonenkis y otros. Los trabajos iniciales estuvieron orientados hacia el reconocimiento de patrones ópticos; sin embargo, excelentes desempeños fueron también obtenidos en regresión y predicción de series de tiempo (Alex & Bernhard, 2004).

La idea básica de las máquinas de soporte vectorial para regresión se explica a continuación.

Suponga que se tienen datos de entrenamiento $(x_1, y_1), ..., (x_l, y_l) \subset X \times \mathbb{R}$, donde X denota el espacio de los patrones de entrada, por ejemplo, $\mathbb{R}^d$. En regresión ε-SV, Vapnik plantea que el objetivo es encontrar una función $f(x)$ que posea como máximo una desviación de ϵ de los objetivos y_i realmente obtenidos, para todos los datos de entrenamiento, y al mismo tiempo tan *suave* como sea posible (Alex & Bernhard, 2004; Vapnik, 2000). En las sub-secciones siguientes se muestra que para lograr este objetivo, matemáticamente el problema se plantea como la búsqueda de una función que sea concebida como el producto escalar del vector de patrones x con un vector desconocido w, el cual debe poseer una norma euclideana mínima.

2.1.1. Caso lineal

Para el caso de que $f(x)$ sea una función lineal f toma la siguiente forma:

$$f(x) = \langle w, x \rangle + b, \; con \; w \in X, \; b \in \mathbb{R} \tag{2.1}$$

donde $\langle \cdot, \cdot \rangle$ denota el producto punto en X. *Suavidad* en el caso de la ecuación (2.1) significa que se debe procurar un w pequeño. Una forma de lograr este objetivo es minimizar la norma euclideana $\|w\|^2$. Formalmente se puede describir este problema como un problema de optimización convexa tal como sigue:

$$\begin{array}{ll} \text{minimizar} & \frac{1}{2}\|w\|^2 \\ \text{sujeto a} & \begin{cases} y_i - \langle w, x_i \rangle - b \leqq \varepsilon \\ \langle w, x_i \rangle + b - y_i \leqq \varepsilon. \end{cases} \end{array} \tag{2.2}$$

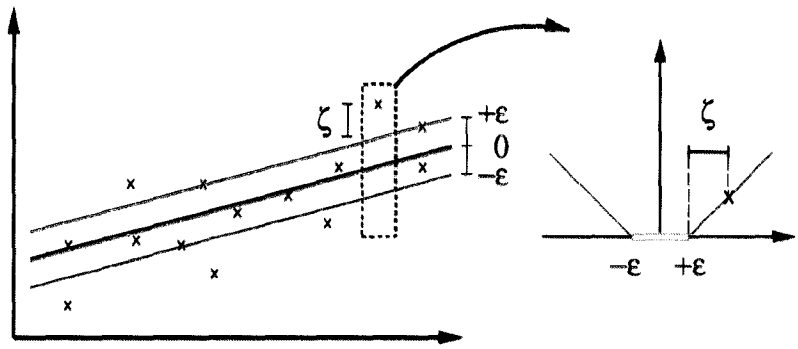

Figura 2.1: Representación gráfica de las variables de holgura en una máquina de soporte vectorial lineal

Fuente: Alex & Bernhard (2004)

2.1.1.1. Variables de holgura

La suposición tácita realizada hasta ahora, y expresada matemáticamente en la ecuación (2.2) es que el problema de optimización convexa es factible. Para solventar este problema se introducen las variables de holgura ξ_i, ξ_i^* (Alex & Bernhard, 2004; Gunn, 1998). En tal sentido, según Vapnik (2000) la formulación (2.2) se convierte en:

$$\begin{aligned} &\text{minimizar} \quad \tfrac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*) \\ &\text{sujeto a} \quad \begin{cases} y_i - \langle w, x_i\rangle - b \leqq \varepsilon + \xi_i \\ \langle w, x_i\rangle + b - y_i \leqq \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \qquad\quad \geqq 0. \end{cases} \end{aligned} \tag{2.3}$$

La constante $C > 0$ determina el equilibrio entre la *suavidad* de f y la tolerancia a desviaciones de ε. La formulación previamente presentada corresponde al tratamiento con la llamada función de pérdida ε-insensible $|\xi|_\varepsilon$ descrita por:

$$|\xi|_\varepsilon = \begin{cases} 0 & \text{si } |\xi| \leq \varepsilon \\ \zeta = |\xi| - \varepsilon & \text{en otro caso.} \end{cases} \tag{2.4}$$

En la figura 2.1 se puede observar gráficamente el comportamiento de las variables de holgura. Sólo los puntos fuera del área sombreada contribuyen al costo, donde las desviaciones generan un comportamiento lineal en $|\xi|_\varepsilon$ representado por el parámetro ζ (Alex & Bernhard, 2004).

2.1.1.2. Formulación dual

El problema de optimización descrito en la ecuación (2.3) pueden ser solucionado con mayor facilidad en su formulación dual. Además, la formulación dual provee la clave para extender las máquinas de soporte vectorial al caso no-lineal. La idea central es construir una función lagrangiana a partir de la función objetivo y las correspondientes restricciones, a través de la introducción de un conjunto dual de variables. También se puede mostrar que esta funcion lagrangiana posee un punto silla con respecto a las variables que describen la función objetivo y las variables duales (Alex & Bernhard, 2004). La función lagrangiana luce de la siguiente forma:

$$L := \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*) - \sum_{i=1}^{l}\alpha_i(\varepsilon + \xi_i - y_i + \langle w, x_i\rangle + b) \tag{2.5}$$

$$-\sum_{i=1}^{l}\alpha_i^*(\varepsilon + \xi_i^* + y_i - \langle w, x_i\rangle - b) - \sum_{i=1}^{l}(\eta_i\xi_i + \eta_i^*\xi_i^*).$$

En este caso, las variables duales, también denominadas multiplicadores de Lagrange, deben satisfacer la restricción de positividad, esto es, $\alpha_i, \alpha_i^*, \eta_i, \eta_i^* \geqq 0$. Por otra parte, tomando en cuenta la condición de punto silla, en un óptimo las derivadas parciales de L con respecto a las variables w, b, ξ_i, ξ_i^* deben hacerse cero.

$$\partial_w L = w - \sum_{i=1}^{l}(\alpha_i - \alpha_i^*)x_i = 0 \tag{2.6}$$

$$\partial_b L = \sum_{i=1}^{l}(\alpha_i^* - \alpha_i) = 0 \tag{2.7}$$

$$\partial_{\xi_i^{(*)}} L = C - \alpha_i^{(*)} - \eta_i^{(*)} = 0 \tag{2.8}$$

Despejando las ecuaciones (2.6), (2.7) y (2.8), y substituyendo variables enen la ecuación (2.5) se obtiene el problema de optimización dual que se muestra en la ecuación (2.9).

$$\text{maximizar} \quad \begin{cases} -\frac{1}{2}\sum_{i,j=1}^{l}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\langle x_i, x_j\rangle \\ -\varepsilon\sum_{i=1}^{l}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{l} y_i(\alpha_i - \alpha_i^*) \end{cases} \tag{2.9}$$

$$\text{sujeto a} \quad \begin{cases} \sum_{i=1}^{l}(\alpha_i - \alpha_i^*) = 0 \\ \alpha_i, \alpha_i^* \in [0, C] \end{cases}$$

En la construcción de la ecuación (2.9) se eliminan las variables η_i, η_i^* a través de la condición (2.8). Así, la ecuación (2.6) puede ser re-escrita como

$$w = \sum_{i=1}^{l}(\alpha_i - \alpha_i^*)x_i \tag{2.10}$$

y por lo tanto

$$f(x) = \sum_{i=1}^{l}(\alpha_i - \alpha_i^*)\langle x_i, x\rangle + b. \tag{2.11}$$

Esta es la llamada *expansión en Vectores Soporte*, donde w puede estar completamente descrito como una combinación lineal de los patrones de entrenamiento x_i (Alex & Bernhard, 2004).

2.1.1.3. Cálculo de b

Según Alex & Bernhard (2004) el cálculo de b puede realizarse aprovechando las condiciones de Karush-Kuhn-Tucker (KKT), las cuales establecen que en la solución óptima el producto entre las variables duales y las restricciones debe hacerse cero. En el caso del soporte vectorial esto significa:

$$\alpha_i(\varepsilon + \xi_i - y_i + \langle w, x_i\rangle + b) = 0 \tag{2.12}$$

$$\alpha_i^*(\varepsilon + \xi_i^* + y_i - \langle w, x_i\rangle - b) = 0$$

y

$$(C - \alpha_i)\xi_i = 0 \tag{2.13}$$

$$(C - \alpha_i^*)\xi_i^* = 0.$$

Las ecuaciones (2.12) y (2.13) permiten realizar algunas conclusiones útiles. En principio, sólo muestras (x_i, y_i) con un correspondiente $\alpha_i^{(*)} = C$ se encuentran fuera del área sombreada del la figura 2.1. A esta región sombreada también suele llamarse "tubo" ε-insensible alrededor de f. Además, $\alpha_i \alpha_i^* = 0$, esto es, no puede haber un conjunto de variables duales α_i, α_i^* que sean simultáneamente diferentes de cero ya que esto requeriría holguras diferentes de cero en ambos sentidos. Finalmente para $\alpha_i^{(*)} \in (0, C)$ se tiene $\xi_i^{(*)} = 0$ y además el segundo factor en la ecuación (2.12) debe desaparecer. Por lo tanto b puede ser calculada tal como sigue:

$$b = y_i - \langle w, x_i \rangle - \varepsilon \quad \text{para } \alpha_i \in (0, C) \tag{2.14}$$

$$b = y_i - \langle w, x_i \rangle + \varepsilon \quad \text{para } \alpha_i^* \in (0, C)$$

2.1.2. Caso no lineal

Existen muchos problemas de regresión/clasificación, donde no es posible realizar estimaciones adecuadas partiendo de la idea de que la función $f(x)$ mencionada en la sección 2.1 pueda tener la forma mostrada en la expresión (2.1). Geométricamente, considerando como ejemplo la clasificación binaria, esta situación puede ser interpretada como la imposibilidad de conseguir una hiper-recta que pueda separar adecuadamente los datos en dos conjuntos diferentes. En estos casos se dice que los datos no son linealmente separables en el espacio de entradas de x, sin embargo, estos mismos datos podrían tener esa característica de separabilidad en un espacio de características dimensionalmente superior dado una "asignación" adecuada $x \mapsto \phi(x)$ (Fletcher, 2009). En la figura 2.2 se ilustra geométricamente el caso en discusión para un problema de clasificación binaria. Inicialmente el conjunto datos no es linealmente separable, sin embargo, después de una re-asignación adecuada, es separable en el espacio de características no-lineal definido implícitamente por una función de base radial, la cual será discutida en secciones posteriores.

Para este caso, denominado "caso no-lineal", la ecuación (2.11) se puede generalizar de la forma siguiente:

$$f(x) \;=\; \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) K(x_i, x) + b \tag{2.15}$$

donde $K(x_i, x) := \langle \phi(x_i), \phi(x) \rangle$ es denominado el núcleo o *kernel* como mejor es conocido y $\Phi(\cdot)$ representa la función que realiza una transformación no lineal de los

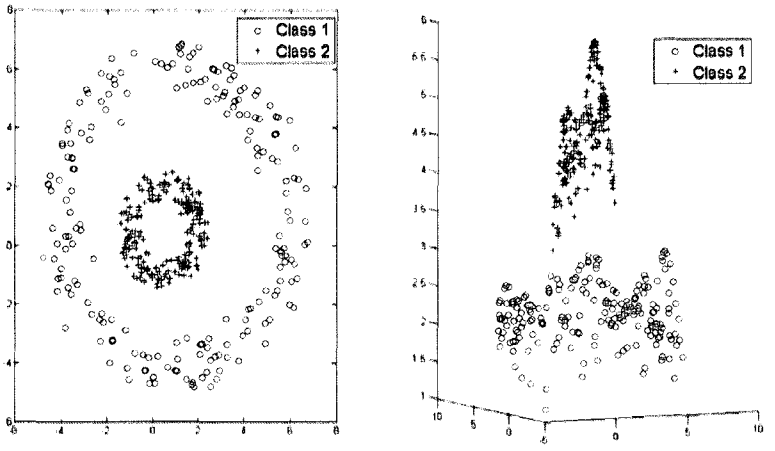

Figura 2.2: Datos dicotómicos re-asignados utilizando una función adecuada $x \mapsto \phi(x)$

Fuente: Fletcher (2009)

patrones de entrada en un *espacio de características* $\mathcal{F}$ dimensionalmente superior. Es importante notar que el producto punto planteado en el caso-lineal no es más que una simplificación del caso no-lineal donde la función $\phi(\cdot)$ no realiza ninguna transformacion en los patrones de entrada.

El reto que ahora surge es encontrar la función $K(x_i, x)$ correspondiente al producto escalar en algún espacio de características $\mathcal{F}$ (Alex & Bernhard, 2004). Para ello es necesario que la función K sea definida positiva y simétrica y además cumpla con las condiciones establecidas en el teorema de Mercer (Vapnik, 2000; Gunn, 1998). En términos precisos se tiene que para garantizar que la ecuación

$$K(x_i, x) = \langle \phi(x_i), \phi(x) \rangle \tag{2.16}$$

se cumpla, es necesario que se cumplan además las siguientes condiciones,

$$K(x_i, x) = \sum_{m}^{\infty} a_m \phi_m(x_i) \phi_m(x), \quad a_m \geq 0, \tag{2.17}$$

$$\int \int K(x_i, x) g(x_i) g(x) dx_i dx > 0, \quad g \in L_2. \tag{2.18}$$

Así, se asegura que el *kernel* represente el producto interno en el espacio de características $\mathcal{F}$.

En el trabajo realizado por Gunn (1998) se puede observar diversos *kernel* que satisfacen las condiciones indicadas en las ecuaciones (2.17) y (2.18) que pueden ser

utilizados en casos particulares. Entre los comúnmente utilizados se puede mencionar los de base radial, los polinomiales y los sigmoidales (Fletcher, 2009). A continuación se describe cada uno de ellos con mayor detalle.

2.1.2.1. Mapeos polinomiales

Un mapeo polinomial es un método popular para el modelado no-lineal ya que ha demostrado un buen desempeño en problemas de reconocimiento de patrones, aproximación de funciones y estimación de regresiones Vapnik (2000). Los mapeos polinomiales poseen formas como

$$K(x, x') = \langle x, x' \rangle^p \tag{2.19}$$

$$K(x, x') = (\langle x, x' \rangle + c)^p \tag{2.20}$$

donde $p \in \mathbb{N}$ y $c > 0$.

La primera expresión suele ser llamada núcleo polinomial homogéneo, mientras que la segunda expresión se denomina núcleo polinomial no-homogéneo (Alex & Bernhard, 2004).

2.1.2.2. Funciones de base radial

Las funciones de base radial, al igual que los mapeos polinomiales, han demostrado un buen desempeño en tareas de reconocimiento de patrones, aproximación de funciones y estimación de regresiones por lo que han recibido especial atención. Según Vapnik (2000) la formulación general de un *kernel* de base radial es

$$K(x, x') = K(|x - x'|). \tag{2.21}$$

Sin embargo, es muy común observar la forma Gausiana de este *kernel* dada por la expresión

$$K(x, x') = exp(-\frac{\|x - x'\|^2}{2\sigma^2}) \tag{2.22}$$

la cual generalmente es llamada *kernel* de base radial o RBF por sus siglas en el idioma inglés (Fletcher, 2009).

Una función de base radial de la forma

$$K(x, x') = exp(-\frac{\|x - x'\|}{2\sigma^2}) \tag{2.23}$$

produce una solución lineal a trozos la cual puede resultar atractiva cuando el problema a resolver acepta discontinuidades en la salida del *kernel* (Gunn, 1998). La función (2.23) es conocida como *kernel* exponencial de base radial.

Las las máquinas de aprendizaje clásicas que utilizan funciones de base radial como las redes neuronales, utilizan agrupamiento de medias-k y los pesos son determinados utilizando propagación hacia atrás (*backpropagation*) (Bernhard *et al.* , 1997). Una característica atractiva de las MSV es que esta selección es implícita, con cada vector soporte contribuyendo a una función gausiana local, centrada en el punto de datos (Gunn, 1998). En otras palabras, el algoritmo de soporte vectorial automáticamente determina los centros, pesos y umbrales para minimizar el límite superior de los errores de validación esperados (Bernhard *et al.* , 1997).

2.1.2.3. Funciones sigmoidales

Un *kernel* que podría parecer admisible, es decir, que cumpla con las condiciones de Mercer, es el tangente hiperbólico ya que se han obtenido muy buenos resultados en su aplicación en funciones de tipo Redes Neurales. La forma general de este tipo de *kernel* es

$$K(x, x') = \tanh(\upsilon + \phi \langle x, x' \rangle). \tag{2.24}$$

Sin embargo, según un análisis realizado por Alex & Bernhard (2004) se muestra que para $\upsilon < 0$ ó $\phi < 0$ este *kernel* no satisface las condiciones establecidas en el teorema de Mercer.

2.1.3. Los vectores soporte

Es importante destacar que de la ecuación (2.12) se puede obtener gran cantidad de información que ayuda a comprender la naturaleza de la expansión en vectores soporte. Inicialmente se tiene que sólo para el caso $|f(x_i) - y_i| \geqslant \varepsilon$ los multiplicadores de Lagrange pueden ser diferentes de cero, o en otras palabras, para todas las muestras dentro del tubo ε-insensible (la región sombreada en la figura 2.1) los multiplicadores α_i, α_i^* desaparecen.

En ese mismo orden de ideas, se tiene que para el caso $|f(x_i) - y_i| < \varepsilon$ el segundo factor en la ecuación (2.12) es diferente de cero, por lo que α_i, α_i^*debe ser cero de tal forma que las condiciones de optimalidad de Karush-Kuhn-Tucker se satisfagan. Además se tiene una expansión dispersa de w en términos de x_i, es decir, no se requiere

de todos los x_i para describir w. Los casos de x_i donde se tienen coeficientes α_i, α_i^* diferentes de cero son los llamados vectores soporte (Alex & Bernhard, 2004).

2.2. El *software* R

La puesta en marcha de la metodología previamente descrita para encontrar funciones basadas en máquinas de soporte vectorial, representa un arduo trabajo de cálculos minuciosos. Afortunadamente, en la comunidad científica internacional existen grandes aportes hacia el desarrollo de *software* libre para cómputo intensivo, que incluyen bibliotecas para la realización de cálculos en diversas áreas de la matemática y la estadística. El aprendizaje de máquina es una de esas áreas donde se han desarrollado paquetes orientados a facilitar la implementación de funciones basadas en esta teoría. En este trabajo se plantea la utilización de R, un *software* libre para cómputo matemático/estadístico, con el objeto de crear las funciones predictoras deseadas.

2.2.1. Características generales de R

El *software* R consiste en un lenguaje y en un ambiente de tiempo de ejecución que ofrece una amplia variedad de funciones estadísticas (modelos lineales y no lineales, pruebas estadísticas clásicas, análisis de series temporales, clasificación, agrupamiento, etc.) y técnicas gráficas, y es altamente extensible ya que posee la habilidad para correr programas almacenados en archivos *script* (Rizzo, 2008).

R es un proyecto GNU desarrollado inicialmente por Robert Gentleman y Ross Ihaka del Departamento de Estadística de la Universidad de Auckland en 1993. Este proyecto es similar al ambiente y lenguaje S desarrollado en los Laboratorios Bell (antes AT & T, ahora Lucent Technologies). R puede ser considerado como una implementación diferente de S. Hay algunas diferencias importantes, sobre todo el hecho de que es un *software* libre disponible de forma gratuita a través de internet, sin embargo desde el punto de vista práctico mucho código escrito para S funciona sin alteraciones en R (Team, 2008).

El término "ambiente" se usa con el objeto de caracterizar R como un sistema totalmente planificado y coherente, en lugar de una acumulación incremental de herramientas muy específicas y poco flexibles, como es frecuentemente el caso con otros programas de análisis de datos. Comúnmente se piensa que R es un sistema de estadísticas, sin embargo, sus desarrolladores prefieren pensar en él como un entorno en el que las técnicas estadísticas son aplicadas (Team, 2008).

R puede ampliarse con facilidad a través de paquetes. Alrededor de ocho paquetes se suministran con su distribución, mientras que miles de paquetes desarrollados como contribuciones voluntarias de expertos estadísticos a nivel mundial están disponibles a través de la familia CRAN de sitios de Internet y cubren una gama muy amplia de la estadística moderna (Team, 2008). Para el desarrollo de este trabajo se utilizaron tres paquetes de R: Kernlab, Rcpp y RInside.

El paquete Kernlab proporciona un conjunto de funcionalidades que permite, entre otros cálculos, crear máquinas de soporte vectorial y realizar predicciones con ellas. Es importante recordar que estos modelos basados en MSV deben ser imbuidos como parte de las creencias de un agente inteligente, es decir, es necesario disponer de ellos para ser utilizados en una aplicación diferente de R. En tal sentido, se propuso encapsular R para C++ utilizando los paquetes Rcpp y RInside, los cuales permiten utilizar R como motor de cálculo para la construcción de funciones basadas en MSV y la realización de predicciones, mientras que el resto de la aplicación se desarrolla en C++.

En las próximas sub-secciones se ofrecen detalles respecto de los paquetes previamente mencionados.

2.2.2. El paquete Kernlab

Kernlab es un paquete desarrollado para R (versión $\geqq$ 2.10) que contiene métodos de aprendizaje de máquina basados en Kernel para clasificación, regresión, agrupamiento, detección de novedades, regresión *cuantil* y reducción de la dimensionalidad. Entre otros métodos, Kernlab incluye máquinas de soporte vectorial, agrupamiento espectral, análisis de componentes principales basados en *kernel* y un solucionador de programación cuadrática (Karatzoglou *et al.* , 2011).

La implementación de máquinas de soporte vectorial realizada en Kernlab, *ksvm*, se basa en los optimizadores encontrados en el paquete bsvm y libsvm disponibles para R. Estos paquetes incluyen una versión muy eficiente de la técnica de optimización por minimización secuencial (SMO, por sus siglas en inglés). La técnica de SMO descompone el problema de programación cuadrática (PC) implícito en las formulación de las MSV evitando utilizar optimización vía PC numérica en alguno de los pasos de la descomposición (Karatzoglou *et al.* , 2011).

2.2.3. Los paquetes Rcpp y RInside

El paquete Rcpp ofrece clases C +− que facilitan enormemente la interfaz C o C ++ en los paquetes de R utilizando la interfaz `.Call()` proporcionada por R. Rcpp provee clases C −− para el vínculo con un gran número de tipos de datos básicos R. Por lo tanto, un autor de algún paquete puede guardar sus datos en estructuras de datos normales de R sin tener que preocuparse acerca de la traducción o la transferencia hacia C −−. Al mismo tiempo, se puede acceder fácilmente a las estructuras de datos en el nivel de C −+, donde luego se pueden manipular sin ninguna restricción. El manejo de los tipos de datos funciona en ambas direcciones. Es tan sencillo pasar los datos de R a C −+, como de C −+ a R (Eddelbuettel & François, 2011).

El paquete RInside es un paquete basado en Rcpp que ofrece clases C +− que hacen que sea más fácil encapsular R en código C −+ ya sea en Linux, OS X o Windows (Eddelbuettel & François, 2010).

La tarea de encapsular R, es decir, hacerlo disponible como un módulo para ser utilizado en otras aplicaciones, se describe en primer lugar en el manual "Escribiendo extensiones de R" disponible en las fuentes principales de esta aplicación. En este manual se describe un procedimiento detallado para realizar este encapsulado, sin embargo, su comprensión requiere de un nivel avanzado en el manejo de las estructuras que componen el código fuente de R. Es por ello que con la finalidad de ofrecer una vía más fácil para utilizar este *software* como motor de cálculo en aplicaciones construidas en C+−, se construyó un conjunto de clases C+− dedicadas a encapsular objetos de R (Eddelbuettel & François, 2010).

2.3. Modelando mezclas de gasolina con MSV

Tal como se mencionó en la sección introductoria de este capítulo, uno de los objetivos de este trabajo es realizar un modelado más preciso del proceso de mezclas de gasolina utilizando máquinas de soporte vectorial. Para ello se cuenta con información operacional que describe el conjunto de datos de entrenamiento $(x_1, y_1), ..., (x_l, y_l)$ mencionado en la sección 2.1 de este capítulo. Cada patrón (x, y) corresponde a una receta x con una propiedad de mezcla asociada y. Las propiedades de mezcla a considerar, tal como se establece en la sección 1.1, son RON, MON, RVP y D-86. Tomando en cuenta que D-86 es una propiedad definida por cinco (5) variables, se definen ocho (8) conjuntos de datos de entrenamiento, por lo que debe ser creada la misma cantidad de funciones predictoras.

Una vez identificados los datos se debe construir cada una de las MSV y para ello se debe resolver el problema de optimización convexa mostrado en la ecuación (2.3). Sin embargo, como el problema abordado es no-lineal, se debe considerar la selección de un *kernel* que permita la transformación no lineal de los patrones de entrada en un espacio de características dimensionalmente superior, tal como se expone en la sección 2.1.2. Es por ello que el desarrollo de la formulación dual mostrada en la sección 2.1.1.2 luce de la siguiente forma

$$maximizar \begin{cases} -\frac{1}{2}\sum_{i,j=1}^{l}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)K(x_i, x_j) \\ -\varepsilon\sum_{i=1}^{l}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{l} y_i(\alpha_i - \alpha_i^*) \end{cases} \tag{2.25}$$

$$sujeto\, a \begin{cases} \sum_{i=1}^{l}(\alpha_i - \alpha_i^*) = 0 \\ \alpha_i, \alpha_i^* \in [0, C]\,. \end{cases}$$

Por lo tanto, tomando en cuenta que

$$w = \sum_{i=1}^{l}(\alpha_i - \alpha_i^*)\phi(x_i) \tag{2.26}$$

la expansión de f puede ser escrita como

$$f(x) = \sum_{i=1}^{l}(\alpha_i - \alpha_i^*)K(x_i, x) + b. \tag{2.27}$$

La ecuación (2.27) se corresponde con la ecuación (2.15) presentada en la sección 2.1.2.

Es importante destacar que la completa definición de cada una de las funciones predictoras, una vez seleccionado un *kernel* adecuado, se resume en calcular tanto los parámetros α_i y α_i^* a través de la resolución del problema de optimización mostrado en la expresión (2.25), como el parámetro b cuya estrategia de cálculo se describe en la sección 2.1.1.3.

Tomando en cuenta los resultados obtenidos en el trabajo realizado por Bernhard *et al.* (1997) el *kernel* a utilizar en este trabajo será el RBF cuya expresión matemática se muestra en la ecuación (2.22). Así, la ecuación (2.27) se convierte en

$$f(x) = \sum_{i=1}^{l}(\alpha_i - \alpha_i^*)exp(-\frac{\|x_i - x\|^2}{2\sigma^2}) + b. \tag{2.28}$$

En términos precisos del planteamiento realizado en este trabajo se define

$$\psi^{(p)}(x) = \sum_{i=1}^{l}(\alpha_i^{(p)} - \alpha_i^{(p)*})exp(-\frac{\|x_i - x\|^2}{2\sigma^2}) + b^{(p)} \tag{2.29}$$

como el modelo basado en MSV de cada una de las propiedades de interés, donde $p = \{\text{RON, MON, RVP, IBP, T10, T50, T90, FBP}\}$.

Ahora bien, el cálculo de los parámetros $\alpha_i^{(p)}$, $\alpha_i^{(p)*}$ y $b^{(p)}$ se llevará a cabo utilizando la aplicación para cálculos estadísticos R y el paquete Kernlab disponible en la misma aplicación.

En el algoritmo 2.1 se muestra un código en lenguaje R para la construcción de funciones predictoras basadas en máquinas de soporte vectorial de cada una de las propiedades de interés.

Código 2.1 Construcción de modelos de mezcla basados en MSV utilizando R

```
1  library(kernlab)
2
3  # Lectura de datos de entrenamiento
4  datos=read.csv(file="/ruta_al_archivo/datos.csv", header=TRUE)
5
6  # Selección del conjunto de datos independientes
7  datos_indep=datos[,1:r]
8
9  # Data frame requeridos en el entrenamiento
10 datos.RON=data.frame(datos_indep,datos$RON)
11 datos.MON=data.frame(datos_indep,datos$MON)
12 datos.RVP=data.frame(datos_indep,datos$RVP)
13 datos.IBP=data.frame(datos_indep,datos$IBP)
14 datos.T10=data.frame(datos_indep,datos$T10)
15 datos.T50=data.frame(datos_indep,datos$T50)
16 datos.T90=data.frame(datos_indep,datos$T90)
17 datos.RVP=data.frame(datos_indep,datos$FBP)
18
19 # Entrenamiento de las máquinas de aprendizaje
20 funcion.RON=ksvm(x=datos.RON.~.,data=datos.RON,kernel="rbfdot")
21 funcion.MON=ksvm(x=datos.MON.~.,data=datos.MON,kernel="rbfdot")
22 funcion.RVP=ksvm(x=datos.RVP.~.,data=datos.RVP,kernel="rbfdot")
23 funcion.IBP=ksvm(x=datos.IBP.~.,data=datos.IBP,kernel="rbfdot")
24 funcion.T10=ksvm(x=datos.T10.~.,data=datos.T10,kernel="rbfdot")
25 funcion.T50=ksvm(x=datos.T50.~.,data=datos.T50,kernel="rbfdot")
26 funcion.T90=ksvm(x=datos.T90.~.,data=datos.T90,kernel="rbfdot")
27 funcion.FBP=ksvm(x=datos.FBP.~.,data=datos.FBP,kernel="rbfdot")
```

En el algoritmo anterior incialmente se hace un llamado al paquete Kernlab a través del comando `library("kernlab")`. Posteriormente se realiza la lectura y almacenamiento de los datos de entrenamiento en la variable `datos` a través de la función `read.csv()`. Ahora `datos` es un *data frame* de tamaño l x m donde l es el

número de muestras y m es la suma del número del número de variables (r) y el número de propiedades (p) asociado a cada receta. Una vez guardados los datos totales de entrenamiento se procede a la construcción de los *data frame* requeridos para el entrenamiento de cada una de las funciones predictoras. Cada uno de estos *data frame* debe estar compuesto por las variables independientes (recetas) y por la propiedad de interés asociada a cada una de estas recetas. Por último se construyen las funciones predictoras utilizando la función $ksvm()$, y como argumentos de estas funciones el *data frame* correspondiente a la propiedad, el nombre de la variable dependeiente en el *data frame* y el tipo de *kernel*.

Detalles de Kernlab y la función $ksvm()$ se pueden encontrar en la documentación asociada al paquete (Karatzoglou *et al.* , 2011).

Capítulo 3

Optimización con restricciones especiales

El método de modelado por minería de datos planteado en el capítulo 2 para la construcción de modelos de mezcla, produce funciones de regresión lineal de alta dimensionalidad, que son difíciles de manejar analíticamente para obtener de ellas sus derivadas y segundas derivadas. Los métodos de optimización citados en el capítulo 1 (sección 1.3.2) dependen de las primeras y segundas derivadas parciales de las restricciones del problema de optimización, por lo que su utilización se limita cuando estas están basadas en modelos difíciles de manejar analíticamente, caso que se presenta en este trabajo.

Para abordar este tipo de situaciones diversos autores han planteado soluciones que han sido denominadas "optimización libre de derivadas" (*derivative free optimization*). En el trabajo realizado por Rios y Sahidinis (Ríos & Sahidinis, 2009), se realiza una comparación de diversos programas que implementan algoritmos para optimización libre de derivadas. De las aplicaciones evaluadas solo existen dos licenciadas como *software* libre: DAKOTA y NOMAD (Le Digabel, 2011). A partir del estudio previamente citado se puede afirmar que NOMAD muestra un desempeño ligeramente superior al DAKOTA, por lo que se escogió esta aplicación para resolver el problema de optimización planteado en esta tesis.

3.1. Formulación matemática general de NOMAD

NOMAD es un *software* que implementa el algoritmo MADS para optimización bajo restricciones generales no-lineales. Este tipo de optimización, también conocida como "optimización libre de derivadas" u "optimización caja negra" (*black box optimization*), trata la optimización de funciones que normalmente están dadas como

programas con gran gasto computacional, sin información referente a las derivadas o sin retorno de valores para un gran número de llamadas. Las restricciones pueden ser de diversa naturaleza, incluso funciones cuyas derivadas son desconocidas, funciones no definidas en todos los puntos, desigualdades no-lineales o dominios discretos (Le Digabel, 2011).

NOMAD está diseñado para solucionar problemas de optimización de la forma

$$\min_{x\in\Omega} f(x) \tag{3.1}$$

o problemas de optimización bi-objetivo como

$$\min_{x\in\Omega}(f_1(x),\ f_2(x)) \tag{3.2}$$

donde $\Omega = \{x \in X \mid (j \in J)\,(c_j(x) \leqslant 0)\} \subset \mathbb{R}^n, f, f_1, f_2, c_j : X \to \mathbb{R} \cup \{\infty\}$ para todos los $j \in J = \{1, 2, ..., m\}$, y X es un subconjunto de $\mathbb{R}^n$. La variable x puede también ser entera, binaria o categórica.

Las funciones f, f_1, f_2 y c_j que definen el problema, son típicamente aproximaciones numéricas a las que no se le puede calcular derivadas y pueden no estar definidas en algunos puntos, incluso algunos que pueden representar soluciones factibles (Le Digabel, 2011).

Una visión general del software NOMAD puede ser apreciada en la figura 3.1.

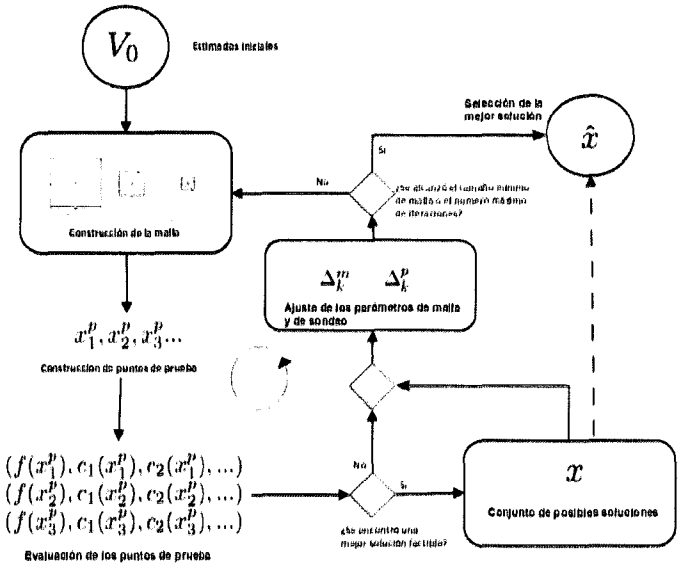

Figura 3.1: Esquema general del software NOMAD para la resolución de problemas de optimización con restricciones generales no-lineales
Fuente: Le Digabel (2011)

El principio general del NOMAD y de cualquier método de búsqueda directa, se basa en el hecho de que el algoritmo iterativamente construye una lista de puntos de prueba x_i^p, que son evaluados por funciones encapsuladas que definen tanto la función objetivo f como las restricciones del problema de optimización c_j (Le Digabel, 2011). Inicialmente el algoritmo considera un estimado inicial o un conjunto de estimados iniciales V_0 a partir del cual se crea el conjunto de puntos de prueba.

3.2. El algoritmo MADS

Dado un estimado inicial $x_0 \in \Omega$, el algoritmo MADS (*Mesh Adaptive Direct Search*) intenta localizar un punto x en Ω para el cual la función f produce un valor mínimo, revisando sistemáticamente el espacio Ω a través de la evaluación f_Ω en algunos puntos de prueba. El algoritmo no requiere ninguna información respecto de la primeras o segundas derivadas de f. Esta condición es útil cuando existen muchos óptimos locales, y es esencial cuando el gradiente de la función ∇f no está disponible ya sea porque no existe o porque no puede ser exactamente estimado debido a señales ruidosas, entre otras razones (Audet & J. E. Dennis, 2006).

MADS es un algoritmo donde en cada iteración un número finito de puntos de prueba se genera y los puntos de prueba no factibles se descartan. La valores de la función objetivo en los puntos de prueba factibles son comparados con el actual valor "titular" $f_\Omega(x_k)$, es decir, el mejor valor de la función objetivo encontrado hasta el momento. Cada uno de esos puntos de prueba se encuentran en la "malla actual", construida de un conjunto finito de n_D direcciones $D \subset \mathbb{R}^n$ escaladas por un "parámetro de tamaño de malla" $\triangle_k^m \in \mathbb{R}_+$ (Audet & J. E. Dennis, 2006).

Existen dos restricciones en el conjunto D. Primero, D debe ser un conjunto compuesto por valores positivos. Segundo, cada dirección $d_j \in D$ (para $j = 1, 2, ..., n_D$) debe ser el producto Gz_j de alguna matriz generadora no-singular fija $G \in \mathbb{R}^{n \times n}$ por un vector de enteros $z_j \in \mathbb{Z}^n$. Por conveniencia, el conjunto D es también visto como una matriz de valores reales $n \times n_D$ (Audet & J. E. Dennis, 2006).

Formalmente la malla se define como

$$M_k = \bigcup_{x \in S_k} \{x + \triangle_k^m Dz : z \in \mathbb{N}^{n_D}\} \tag{3.3}$$

donde S_k, es el conjunto de puntos donde la función objetivo f ha sido evaluada al momento del inicio de la iteración k.

La evaluación de f_Ω en cada punto de prueba x se realiza como sigue: inicialmente, las restricciones que definen Ω son validadas para comprobar si x es o no factible. Luego, si $x \notin \Omega$ entonces $f_\Omega(x)$ es ajustado a $+\infty$ obviando su evaluación, y sin evaluar, además, todas las restricciones que definen Ω (Audet & J. E. Dennis, 2006).

Cada iteración se divide en tres pasos, el sondeo (*poll*), la búsqueda (*search*) y las actualizaciones. El paso de búsqueda es flexible y permite la creación de puntos de prueba sobre la malla. Esa libertad es posible gracias al hecho de que la convergencia del método no depende de este paso, tal como se muestra en el trabajo realizado por Audet y Denis (Audet & J. E. Dennis, 2006). La flexibilidad del paso de búsqueda es aprovechada por NOMAD a través de la definición de estrategias genéricas como el muestreo de Hipercubo Latino (*Latin Hypercube sampling*) o la búsqueda de variables vecinas (*variable neighborhood search*) aunque su utilización no fue necesaria en esta tesis. Además, los usuarios pueden definir sus propias estrategias de búsqueda para aplicaciones específicas (Le Digabel, 2011).

El paso de sondeo está definido de una forma más rígida ya que el análisis de convergencia se basa en él. En este paso se explora la malla en una zona cercana a la iteración actual x_k con el siguiente conjunto de "puntos de prueba de sondeo"

$$P_k = \{x_k + \triangle_k^m d : d \in D_k\} \subset M_k \tag{3.4}$$

donde D_k es el conjunto de direcciones de sondeo. Cada columna de D_k es una combinación entera de columnas de D, y D_k es tal que sus columnas forman un conjunto que contiene valores positivos. Los puntos de P_k se generan tal que su distancia del centro de sondeo x_k es inferiormente limitada por el "parámetro de tamaño de sondeo" $\triangle_k^p \in \mathbb{R}^+$. El algoritmo MADS define el vínculo entre parámetros de tamaño de sondeo y de malla: $\triangle_k^m$ es siempre más pequeño que $\triangle_k^p$ y $\triangle_k^m$ se reduce más rápidamente que $\triangle_k^p$ despues de algún intento fallido. La figura 3.2 ilustra la relación entre estos dos parámetros, donde los t_i representan los puntos de prueba x^p de la figura 3.1.

Los pasos de sondeo y búsqueda generan puntos de prueba sobre la malla. Las funciones luego son evaluadas en esos puntos y el algoritmo determina si cada evaluación es o no exitosa. Al final de la iteración k, un paso de actualización determina el estatus de la iteración (éxito o fracaso) y se escoge el punto de prueba de la próxima iteración x_{k+1}. Esta corresponde al éxito más prometedor o se mantiene en x_k. El tamaño del parámetro de malla es también actualizado a

$$\triangle_{k+1}^m = \tau^{w_k} \triangle_k^m \tag{3.5}$$

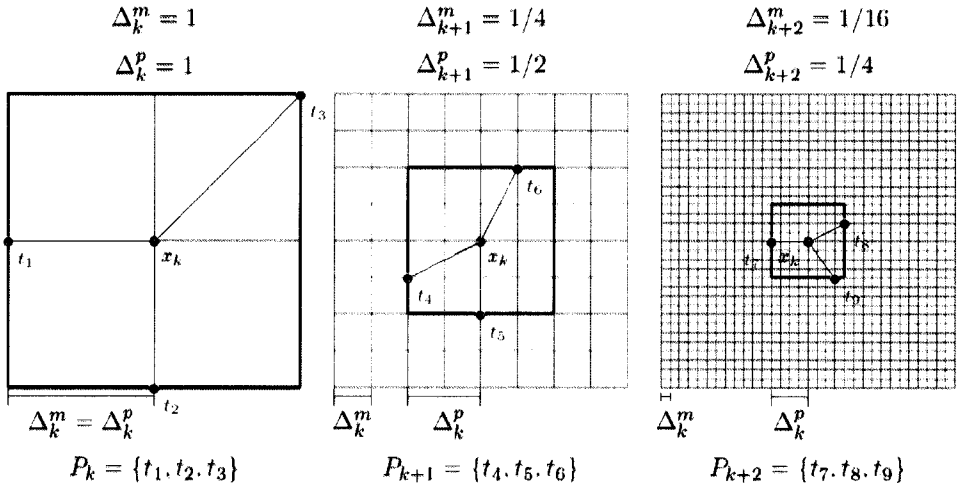

Figura 3.2: Ejemplo de diferentes configuraciones de malla con $n = 2$. Las líneas delgadas representan la malla de tamaño $\triangle_k^m$, y las líneas gruesas los puntos a la distancia $\triangle_k^p$ de x_k. Se ilustran los puntos de sondeo en $n + 1$ direcciones. La red se reduce entre las situaciones para mostrar como $\triangle_k^m$ se reduce más rápidamente que $\triangle_k^p$

Fuente: Le Digabel (2011)

donde $\tau > 1$ es un número racional dado y w_k es un entero finito, positivo o nulo si la iteración k es un éxito, o estrictamente negativo si la iteración falla. En otras palabras el parámetro de tamaño de malla decrece ante intentos fallidos y posiblemente aumenta ante intentos exitosos (Le Digabel, 2011).

El parámetro de tamaño de sondeo también se actualiza en este paso de acuerdo a reglas que dependen de la implementación.

3.3. Optimización utilizando NOMAD y modelos basados en MSV

3.3.1. Formulación matemática del problema de optimización

En el capítulo 1, específicamente en la sección 1.3.2, se muestra la expresión (1.11) que describe matemáticamente el problema de optimización encontrado en el problema de mezclas de gasolina. En esta expresión se define a $g(Q, X)$ como la representación no-lineal de los modelos de mezcla de propiedades.

Luego, en el capítulo 2, específicamente en la sección 2.3, se muestra la ecuación (2.29), que es la expresión matemática resultante del modelaje de las propiedades

de mezcla de combustibles como máquinas de aprendizaje, específicamente, como Máquinas de Soporte Vectorial.

Si se define $\Psi(X)$ como un vector que contiene las predicciones de las propiedades de mezcla en función de una receta X dada, es decir,

$$\Psi(X) = \begin{bmatrix} \psi^{RON}(X) \\ \psi^{MON}(X) \\ \vdots \\ \psi^{FBP}(X) \end{bmatrix} \tag{3.6}$$

se podría sustituir $g(Q, X)$ por $\Psi(X)$ en la expresión (1.11), obteniendo finalmente la formulación

$$\begin{aligned} &\max_{x} (CX) \\ &\Psi(X) \leqq (e^t X)S + \beta \\ &HX \leqq \rho \end{aligned} \tag{3.7}$$

que describe el problema matemático exacto a resolver en este estudio.

Haciendo una analogía entre la expresión (3.7) y la formulación matemática general de NOMAD presentada en la sección 3.1, se podría decir que

$$f(X) = CX \tag{3.8}$$

y a cada $c_j(X)$ le corresponde un elemento del vector

$$\begin{bmatrix} \Psi(X) - (e^t X)S + \beta \\ HX - \rho \end{bmatrix} \leq 0 \tag{3.9}$$

lo que corrobora la factibilidad de utilizar NOMAD para resolver el problema de optimización encontrado en este estudio.

3.3.2. Detalles de implementación de NOMAD

Una vez formulado el problema matemático exacto es posible describirlo en forma codificada planteando su solución a través de la utilización del software NOMAD. Se propuso realizar la implementación de la solución en el lenguaje de programación C++ que por ser el utilizado para desarrollar NOMAD describe un escenario más favorable para este estudio. Como compilador se utiliza GCC, la colección de compiladores de GNU (http://www.gnu.org/).

En el código 3.1 se muestran algunos detalles de la implementación de NOMAD en C++ para la resolución del problema de optimización planteado en la sub-sección anterior.

Código 3.1 Implementación de NOMAD en C++

```
1  // Implementación de NOMAD para la resolución del problema de optimización
2  // de mezclas utilizando funciones basadas en MSV como restricciones.
3  try {
4    NOMAD::begin ( );  // Inicialización de NOMAD
5    // Creación de variable de tipo Parameters:
6    NOMAD::Parameters p ( out );
7      // Nº de variables de salida entre restricciones (m)  y función objetivo (1)
8    vector<NOMAD::bb_output_type> bbot (m+1);
9    // Definición de tipos de salidas
10   bbot[0] = NOMAD::OBJ;
11   bbot[1] = NOMAD::PB;
12        .
13        .
14        .
15   bbot[m] = NOMAD::PB;
16    // Declaración y definición de vector de inicialización del algoritmo
17    // Límites superior e inferior y tamaño mínimo e inicial de red.
18   NOMAD::Point sp ( r );// r: dimensión del vector receta
19   NOMAD::Point lb ( r );  // Límite inferior
20   NOMAD::Point ub ( r );  // Límite inferior
21   NOMAD::Point ims( r );  // Tamaño inicial de malla
22   NOMAD::Point mms( r );// Tamaño mínimo de malla
23   int k;
24   for (k=0;k<r,k++){
25     sp[k] = inicioAlgoritmo[k];// inicioAlgoritmo vector previamente definido
26     lb[k] = limiteInferior[k];// limiteInferior vector previamente definido
27     ub[k] = limiteSuperior[k];// limiteSuperior vector previamente definido
28     ims[k]= tamañoInicial[k];// tamañoInicial vector previamente definido
29     mms[k]= tamañoMinimo[k];// tamañoMinimo vector previamente definido
30   }
31   // Asignaciones de variables en p (Parameters)
32   p.set_DIMENSION (r);// Inicialización de Nomad
33   p.set_BB_OUTPUT_TYPE ( bbot );// Tipos de variables de salida
34   p.set_X0 ( sp );// Inicialización del algoritmo
35   p.set_LOWER_BOUND ( lb );// Límite inferior
36   p.set_UPPER_BOUND ( ub );// Límite superior
37   p.set_MAX_BB_EVAL (numIter);// Número máximo de iteraciones
38   p.set_INITIAL_MESH_SIZE(ims);// Tamaño inicial de la red
39   p.set_MIN_MESH_SIZE(mms);// Tamaño mínimo de la red
40   p.set_SNAP_TO_BOUNDS(false);
41   // Validación de parámetros:
42   p.check();
43   // Creación de función de evaluación que
44   // comprende función objetivo y restricciones:
45   My_Evaluator ev   ( p,...);
46   // Creación y ejecución del algoritmo:
47   NOMAD::Mads mads ( p , &ev );
48   mads.run();
49 }      catch ( exception & e ) {
50 cerr << "\nNOMAD has been interrupted (" << e.what() << ")\n\n";
51 }
52 NOMAD::Slave::stop_slaves ( out );
53 NOMAD::end();
```

En el código anterior se puede identificar que en la línea 4 se hace la inicialización de NOMAD mientras que en la línea 6 se define una variable de tipo `NOMAD::Parameters`.

Esta variable es donde se almacena toda la configuración de la corrida de optimización.

Código 3.2 Declaración de la clase `My_Evaluator`

```
1  class My_Evaluator : public NOMAD::Evaluator {
2  public:
3    My_Evaluator ( NOMAD::Parameters & p) :
4      NOMAD::Evaluator ( p) {
5
6    R_MyEval.parseEvalQ("library(kernlab)");
7
8    // Lectura de datos de entrenamiento
9    R_MyEval.parseEvalQ("datos=read.csv(file=\"/ruta_archivo/datos.csv\")");
10
11   // Selección del conjunto de datos independientes
12   R_MyEval.parseEvalQ("datos_indep=datos[,1:r]");
13
14   // Data frame requeridos en el entrenamiento
15   R_MyEval.parseEvalQ("datos.RON=data.frame(datos_indep,datos$RON)");
16   R_MyEval.parseEvalQ("datos.MON=data.frame(datos_indep,datos$MON)");
17   R_MyEval.parseEvalQ("datos.RVP=data.frame(datos_indep,datos$RVP)");
18   R_MyEval.parseEvalQ("datos.IBP=data.frame(datos_indep,datos$IBP)");
19   R_MyEval.parseEvalQ("datos.T10=data.frame(datos_indep,datos$T10)");
20   R_MyEval.parseEvalQ("datos.T50=data.frame(datos_indep,datos$T50)");
21   R_MyEval.parseEvalQ("datos.T90=data.frame(datos_indep,datos$T90)");
22   R_MyEval.parseEvalQ("datos.RVP=data.frame(datos_indep,datos$FBP)");
23
24   // Entrenamiento de las máquinas de aprendizaje
25   R_MyEval.parseEvalQ("funcion.RON=ksvm(x=datos.RON.~.,data=datos.RON,kernel=\"rbfdot
         \")");
26   R_MyEval.parseEvalQ("funcion.MON=ksvm(x=datos.MON.~.,data=datos.MON,kernel=\"rbfdot
         \")");
27   R_MyEval.parseEvalQ("funcion.RVP=ksvm(x=datos.RVP.~.,data=datos.RVP,kernel=\"rbfdot
         \")");
28   R_MyEval.parseEvalQ("funcion.IBP=ksvm(x=datos.IBP.~.,data=datos.IBP,kernel=\"rbfdot
         \")");
29   R_MyEval.parseEvalQ("funcion.T10=ksvm(x=datos.T10.~.,data=datos.T10,kernel=\"rbfdot
         \")");
30   R_MyEval.parseEvalQ("funcion.T50=ksvm(x=datos.T50.~.,data=datos.T50,kernel=\"rbfdot
         \")");
31   R_MyEval.parseEvalQ("funcion.T90=ksvm(x=datos.T90.~.,data=datos.T90,kernel=\"rbfdot
         \")");
32   R_MyEval.parseEvalQ("funcion.FBP=ksvm(x=datos.FBP.~.,data=datos.FBP,kernel=\"rbfdot
         \")");
33     }
34   ~My_Evaluator ( void ) {}
35   RInside R_MyEval;     // Creación de objeto de tipo RInside
36   bool eval_x ( NOMAD::Eval_Point    & x,
37           const NOMAD::Double & h_max,
38           bool & count_eval);
39 };
```

Luego se declara cada uno de los parámetros que componen la configuración de la corrida que son: número de variables de salida incluyendo la función objetivo y las restricciones (línea 8), tipos de variables de salida (líneas 9 a 15), receta inicial V_0 (línea 18), límite inferior de las variables de decisión (línea 19), límite superior de las variables de decisión (línea 20) y tamaño inicial y mínimo de malla $\triangle_k^m$ (líneas 21 y 22). Una vez declarados estos parámetros se definen sus valores entre las líneas 23

y 30 para luego asignarlos al parámetro `p` entre las líneas 32 y 39. En la línea 40 se especifica que la evaluación de puntos de prueba no pueda ser realizada fuera de los límites establecidos para las variables de decisión.

Si bien hasta este momento ya se ha definido la configuración de los parámetros de la optimización, aún no se ha definido la función que implementa tanto la función objetivo como las restricciones. Estas deben ser implementadas como método de un objeto de tipo `My_Evaluator` que se declara en la línea 45. Por último, en la línea 47 se construye un objeto de tipo `NOMAD::Mads` tomando como argumentos el parámetro `p` y una referencia al objeto `ev` de tipo `My_Evaluator`. La ejecución del algoritmo de optimización se inicia en la línea 48.

En el código 3.2, específicamente en la línea 35, se muestra la declaración de la clase `My_Evaluator` donde se define como atributo un objeto de tipo `RInside` llamado `R_MyEval`. Luego, en el constructor, específicamente entre las líneas 6 y 32, se implementa el código 2.1 para el entrenamiento de los modelos de mezcla de propiedades encapsulando adecuadamente cada una de las líneas de código a través del uso del comando `R_MyEval.parseEvalQ(" ")` provisto por el paquete RInside.

En el código 3.3 se muestran algunos detalles de implementación de la función `My_Evaluator::eval_x()` que es donde se llama a R para realizar el cálculo de la función objetivo y las restricciones en cada una de las iteraciones. Inicialmente, entre las líneas 1 y 9, se declara un conjunto de variables de tipo `NOMAD::Double` donde se almacenan tanto las calidades predichas por los modelos de mezcla entrenados en el constructor del objeto `My_Evaluator::ev`, como los resultados de las restricciones $c_j(x_i^{(p)})$. Luego, entre las líneas 11 y 15, los valores de la receta $x_i^{(p)}$ que representa el i-ésimo punto de prueba se almacenan en variables tipo `double` para posteriormente entre las líneas 16 y 20 ser asignadas a una variable en el objeto `R_MyEval` a través del comando `R_MyEval.assign()`. Entre las lineas 21 y 25 los valores de $x_i^{(p)}$ se almacenan en una estructura tipo *data frame* que después es utilizada entre las líneas 27 y 34, en conjunto con los modelos de mezcla basados en MSV, para predecir las propiedades de mezcla a través de la función `predict()` en el lenguaje R. Una vez calculadas las propiedades de mezcla, se procede a almacenar estos valores en variables de tipo `NOMAD::Double` entre las líneas 36 y 43 para luego definir entre las líneas 45 y 49, y en función de dichos resultados, los valores de $c_j(x_i^{(p)})$ para $j \in J = \{1, 2, ..., m\}$. En la línea 44 se calcula la función objetivo que representa el costo de la mezcla. Finalmente, entre las líneas 50 y 55, se asignan las salidas de la función `eval_x()` a través de la función `x.set_bb_output()` donde se asigna en el elemento 0 el resultado de la

evaluación de la función objetivo y las demás restricciones en el resto de los elementos, tal como se especifica entre las líneas 10 y 15 del código `3.1`.

Código 3.3 Implementación de la función `My_Evaluator::eval_x()`

```
1 NOMAD::Double predictRON;
2 NOMAD::Double predictMON;
3 NOMAD::Double predictRVP;
4 NOMAD::Double predictIBP;
5 NOMAD::Double predictT10;
6 NOMAD::Double predictT50;
7 NOMAD::Double predictT90;
8 NOMAD::Double predictFBP;
9 NOMAD::Double c1,...,cm;
10 // Preparando el conjunto de evaluación
11 double x0 = x[0].value();
12     .
13     .
14     .
15 double xR = x[R].value(); // R=r-1. donde r es el tamano del vector receta
16 R_MyEval.assign( x0,"x0" );
17     .
18     .
19     .
20 R_MyEval.assign( xR,"xR" );
21 R_MyEval.parseEvalQ ("evalData[1,1]<-x0");
22     .
23     .
24     .
25 R_MyEval.parseEvalQ ("evalData[1,r]<-xR");
26 // Evaluando
27 R_MyEval.parseEvalQ("predictRON = predict(funcion.RON,evalData[1,1:r])");
28 R_MyEval.parseEvalQ("predictMON = predict(funcion.MON,evalData[1,1:r])");
29 R_MyEval.parseEvalQ("predictRVP = predict(funcion.RVP,evalData[1,1:r])");
30 R_MyEval.parseEvalQ("predictIBP = predict(funcion.IBP,evalData[1,1:r])");
31 R_MyEval.parseEvalQ("predictT10 = predict(funcion.T10,evalData[1,1:r])");
32 R_MyEval.parseEvalQ("predictT50 = predict(funcion.T50,evalData[1,1:r])");
33 R_MyEval.parseEvalQ("predictT90 = predict(funcion.T90,evalData[1,1:r])");
34 R_MyEval.parseEvalQ("predictFBP = predict(funcion.FBP,evalData[1,1:r])");
35
36 predictRON = R_MyEval["predictRON"];
37 predictMON = R_MyEval["predictMON"];
38 predictRVP = R_MyEval["predictRVP"];
39 predictIBP = R_MyEval["predictIBP"];
40 predictT10 = R_MyEval["predictT10"];
41 predictT50 = R_MyEval["predictT50"];
42 predictT90 = R_MyEval["predictT90"];
43 predictFBP = R_MyEval["predictFBP"];
44 cost_Mezcla = (x0*cost0 + x1*cost1 + ... + xR*costR)/100;
45 c1= ...;//Las restricciones dependen de las predicciones de las propiedades de la
       mezcla
46     .
47     .
48     .
49 cm=...;
50 x.set_bb_output  ( 0 , cost_Mezcla );    // objective value
51 x.set_bb_output  ( 1 , c1 );     // objective value
52     .
53     .
54     .
55 x.set_bb_output  ( m , cm);
56 count_eval = true; // Cuenta una nueva evaluación de la función eval_X
57 return true; // La evaluación fue exitosa
```

3.4. Comentarios finales

En este capítulo se realizó una descripción del algoritmo MADS para la resolución de problemas de optimización bajo circunstancias especiales, donde obtener las derivadas de las funciones que integran la función objetivo y/o las restricciones de la optimización es una tarea muy difícil o imposible.

La descripción realizada ofrece una solución a uno de los problemas abordados en esta tesis, donde se desea realizar la optimización en línea del proceso de mezclas de gasolina, mientras que las mezclas son modeladas utilizando MSV. Si bien las funciones basadas en MSV son algebraicas, tal como se muestra en la Ecuación (2.29), dependiendo del *Kernel* seleccionado la determinación de sus derivadas puede convertirse en una tarea compleja por lo que se estaría en presencia de un caso especial de optimización.

Es por ello que se propone utilizar el software NOMAD, quien implementa el algoritmo MADS, como parte de las funciones incorporadas en el agente inteligente responsable de realizar la OTR para mezclas de gasolina. En este caso, el algoritmo MADS debe formar parte de las funciones de razonamiento del agente, las cuales pueden derivar en nuevas metas, usando las metas y observaciones previas.

Aún considerando que la solución de un problema de optimización donde funciones de la forma de la Ecuación (2.29) formen parte de las restricciones podría usar aquellos métodos basados en derivadas, el enfoque planteado en esta tesis es más robusto, modular y mantenible; y por lo tanto abre el camino a la exploración de funciones cada vez más complejas que puedan no ser derivables o definidas en algunos puntos.

El análisis de la estructura conceptual del agente se muestra con detalles en el capítulo siguiente.

Capítulo 4

Optimización en línea basada en agentes

En el capítulo 1 se estableció que la optimización en tiempo real (OTR) es un enfoque de control de procesos basado en modelos, que utiliza información actual del proceso para predecir las políticas de operación óptima para una unidad de proceso durante el próximo intervalo de OTR (Forbes *et al.* , 2002). En este sentido la OTR del proceso de mezclas representa una oportunidad clave para la minimización de costos en el proceso de refinación.

La OTR ha sido una estrategia de provecho para la solución del problema referente a la obtención de mezclas fuera de especificación (Barsamian, 2003; Forbes *et al.* , 2002; Diaz & Barsamian, 1996; Monder, 2001; Sullivan, 1990; Vermeer *et al.* , 1997). Sin embargo, para obtener un buen desempeño de esta estrategia es muy importante contar con un modelo adecuado del proceso. Generalmente los modelos utilizados en estas aplicaciones son aproximaciones empíricas lo cual podría generar un desempeño no satisfactorio de la estrategia de optimización. Es por ello que el foco de este trabajo es el diseño y desarrollo de un optimizador en línea de mezclas de gasolina, que cuente con modelos que representen de una manera más precisa la mezcla de combustibles, con el objeto de tener estimaciones mas adecuadas de las propiedades de la mezcla final, e incrementar así la eficacia de las estrategia de OTR.

Tal como se muestra en la figura 1.4, los elementos que toman parte en una estrategia de optimización del proceso de mezclado de gasolinas son diversos y pueden ser identificados como: el sistema de control distribuido, que ejecuta la acción de control sobre la planta; el optimizador en línea, que realiza los cálculos a partir de información actualizada; el analizador en línea que determina en línea las propiedades tanto de los básicos como del producto obtenido; y el sistema de información de inventarios (Wang *et al.* , 2007).

Asimismo, todos los elementos de un sistema de mezclas deben ser tomados en cuenta para implementar un sistema para su optimización. La transferencia de datos entre los distintos módulos debe ser automática, no debe ser necesaria la realimentación manual de la misma información repetidas veces, la consideración de escalamientos, ni los cambios de unidades métricas ya que estos detalles redundan en una alta probabilidad de errores humanos incrementando además el tiempo para la ejecución de la tarea (Barsamian, 2003).

En tal sentido, se plantea la necesidad de crear un sistema optimizador en línea como una aplicación *software* que esté concebida para su interacción durante el proceso de optimización, con otros elementos que formen parte del sistema. Considerando que un agente inteligente es un sistema computacional localizado en un entorno y que es capaz de actuar autónoma y flexiblemente en dicho ambiente, de manera tal de alcanzar los objetivos para el cual fue diseñado (Ramírez *et al.* , 2009), se propone que el desarrollo del sistema optimizador en estudio se conciba como un agente inteligente.

4.1. Agentes inteligentes y optimización de procesos

En la industria de procesos químicos la información y los datos son recursos importantes, aparte de los materiales y la energía. La efectiva administración de la variedad de recursos de información se hace necesaria para su mejor acceso e intercambio, lo cual es vital para un desarrollo de productos colaborativos y una manufactura integrada. Sin embargo, la mayoría de los recursos de información técnica y operacional de sistemas de ingeniería química residen en formato electrónico, en un ambiente de red donde, no obstante, no se comunican todos los diferentes elementos que componen dicha red. La mayoría de los productos de *software* utilizados para solucionar problemas operacionales o de diseño de ingeniería no son capaces de comunicarse entre sí lo que hace cuesta arriba la posibilidad de que estos programas puedan mantener algún tipo de cooperación (Gao *et al.* , 2009).

En el trabajo desarrollado por Gao *et al.* (2009) se parte de las premisas descritas en el párrafo anterior para proponer una arquitectura basada en agentes inteligentes que permita integrar la información distribuida en repositorios electrónicos y los distintos *software* utilizados para simulación y soporte a las decisiones basado en reglas, con el objeto de contar con una "manufactura colaborativa" que permita ejecutar de una forma más eficiente tareas de análisis, monitorización y predicción del desempeño de procesos.

En la figura 4.1 se puede observar que en el enfoque planteado por Gao *et al.* (2009), se sugiere una arquitectura donde se disponga tres tipos de agentes inteligentes: agentes usuarios, que reciben requerimientos de los usuarios y despliegan resultados; agentes intermediarios, responsables por la descomposición y coordinación de la ejecución de cada una de las tareas requeridas para satisfacer el requerimiento del usuario; y agentes de tareas que ejecutan actividades específicas asignadas por el agente intermediario. Es importante hacer notar que entre los agentes de tareas descritos en el trabajo de Gao *et al.* (2009) destacan los requeridos para llevar a cabo cálculos de optimización y programación, tal como corresponde al agente propuesto en este trabajo de tesis.

Por otra parte, en el trabajo realizado por Ramírez *et al.* (2009) se propone una arquitectura para supervisión inteligente de procesos basada en sistemas multiagentes. Se plantea la creación de un conjunto de agentes que ejecuten tareas específicas de control considerando sus metas individuales, siendo a su vez todos ellos coordinados por un agente global supervisor. Se destacan las ventajas de la modularidad en el uso de agentes, ya que cada agente de *software* puede ser desarrollado en diferentes plataformas y utilizando diversos lenguajes de programación. Los resultados obtenidos en entornos simulados muestran, además, un buen desempeño global del sistema.

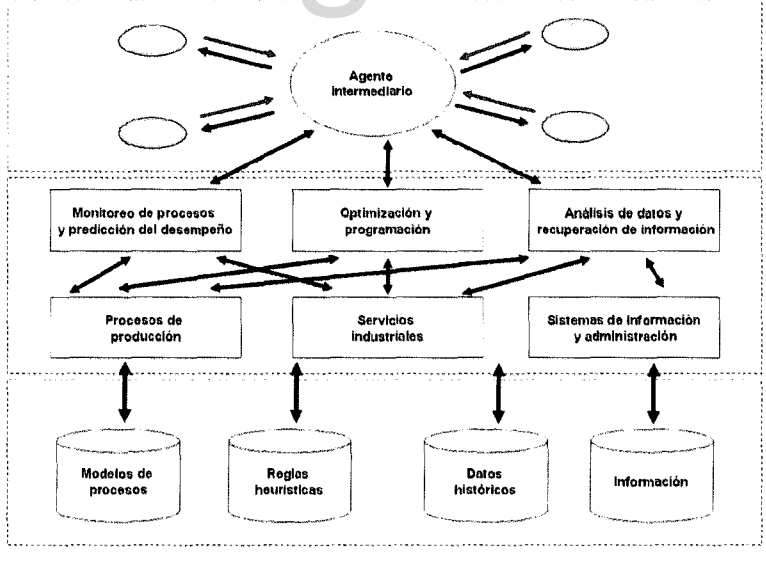

Figura 4.1: Enfoque de sistema multiagentes para el apoyo en la toma de decisiones en la industria química

Fuente: Gao *et al.* (2009)

Finalmente, en el trabajo desarrollado por Gaoa *et al.* (2006) se plantea una

arquitectura orientada a agentes, para modelar y realizar simulación multiobjetivo de un proceso con reacciones químicas. En este trabajo, agentes individuales interactúan de forma asíncrona a través del pase de mensajes entre hilos, donde, uno de ellos tiene un objetivo específico y puede enviar o recibir información convenientemente. Los resultados obtenidos muestran un excelente desempeño de la aplicación en tiempo real.

Estos antecedentes muestran la conveniencia del uso de arquitecturas para optimización y control de procesos basada en agentes inteligentes en la industria de procesos químicos desde dos puntos de vista. En el primero de ellos se destaca la posibilidad de que los agentes traten con recursos de información diversos y heterogéneos, incorporen codigos heredados *(legacy)* encapsulados y funcionen en un ambiente de red distribuida. Además, se hace énfasis en la posibilidad de contar con una plataforma más fácil de mantener y escalar (Gao *et al.* , 2009). En el segundo punto de vista se muestran resultados concretos respecto de la superioridad de sistemas basados en agentes para ejecutar tareas de control y optimización de procesos (Ramírez *et al.* , 2009; Gaoa *et al.* , 2006).

4.2. Definición formal de un agente inteligente

Según Dávila (2011), se define un agente como una 7-tupla

$$\langle P_a, B_a, M_a, percibe, actualiza, planifica, aprende \rangle \tag{4.1}$$

compuesta por los conjuntos: P_a, o perceptos, unidades de percepción que el agente puede capturar de todas las bases de conocimiento posibles B_a, de todas las metas que se pueda plantear un agente M_a, así como las funciones: *percibe*, *actualiza*, $planifica$ y *aprende*

Un agente inteligente posee una base de conocimiento o de creencias, tomada del conjunto de todas las bases posibles B_a y un conjunto de metas (también llamadas intenciones), elemento del conjunto de todas las metas posibles M_a. La metas y las creencias caracterizan conjuntamente el estado interno del agente (Dávila, 2011).

La función *percibe*() describe cómo el agente percibe su entorno y devuelve unidades de percepción. La función *actualiza*() especifica el mecanismo de memoria del agente y debe garantizar que la incorporación de nueva información preserve la estructura interna de la base de conocimiento, así como su consistencia, puesto que esa base es una colección de fórmulas con una sintaxis y una semántica bien definida. La función $planifica()$ representa el razonamiento del agente que puede derivar nuevas

metas e influencias usando las metas y observaciones previas y la base de conocimiento. Por último, la función *aprende*() da cuenta del proceso de aprendizaje que transforma la base de conocimiento actual del agente en una nueva, a partir de una "tarea de aprendizaje" disparada por acciones que fallan en alcanzar alguna meta (Dávila, 2011).

4.3. Un agente inteligente como optimizador de mezclas de gasolina

Una vez descrito el enfoque utilizado previamente para implementar agentes inteligentes como parte de un sistema para la optimización y/o supervisión de un proceso industrial, y habiendo definido formalmente un agente inteligente, se procede a realizar el planteamiento formal del agente propuesto en este estudio.

4.3.1. Enfoque conceptual

Atendiendo al enfoque planteado en la sección 4.2, a continuación se detalla cada uno de los conjuntos y funciones que componen la 7-tupla que definen el agente inteligente a desarrollar para ejecutar la optimización en línea del proceso de mezclas de gasolina.

La base de conocimientos o de creencias B_a abarca el conjunto de observaciones del proceso que incluyen una receta dada y las propiedades de mezcla obtenidas con ese receta.

La función *aprende*() se encarga de la generalización de la base de conocimientos del agente . En términos específicos del agente en desarrollo se debe agregar un nuevo patrón de entrenamiento (x, y) una vez finalizada y caracterizada cada mezcla, y luego, entrenar los modelos de mezcla considerando el nuevo patrón de entrenamiento. La generalización de B_a se propone como la función $predCal()$. En el desarrollo de este trabajo a esta función se le denomina el modelo del proceso, y tal como se expone con detalles en el capítulo 2, se propone su implementación como modelos generados utilizando MSV.

El conjunto de metas M_a corresponde a la misión del agente de mantener el proceso tan económico como sea posible a través de la minimización del costo del barril de mezcla, mientras se cumple con todas las restricciones de calidad del producto. En otras palabras, el agente puede tener solo dos metas, optimizar la mezcla o no optimizar la mezcla, salvo porque el cómo optimizar la mezcla se obtiene con la primera meta.

La función *percibe*() representa la comunicación del agente con el resto de los componentes del sistema. Según la figura 1.4 el agente debe comunicarse con los elementos que componen el proceso de optimización fuera de línea y con los elementos que componen el control básico del proceso. Sin embargo, es con estos últimos con quienes el agente debe tener una comunicación más frecuente ya que son estos elementos los que materializan en el proceso las instrucciones que el agente establece para alcanzar sus metas. Asimismo, los elementos de control básico informan al agente el estado del proceso.

La función *actualiza*() corresponde al conjunto de métodos que vuelven capaz al agente de reconocer la situación actual, agregando en su base de conocimiento las observaciones realizadas en el intento de alcanzar alguna meta.

Por último, $planifica()$ representa el conjunto de funciones que ejecutan la optimización del proceso, y que hacen posible alcanzar las metas del agente. Dentro de la función planifica el agente decide el camino más adecuado para cumplir con la meta que tiene establecida y que corresponde a la ejecución de la mejor mezcla posible en términos de costo y calidad.

4.3.2. Detalles de implementación

En la figura 4.2 se muestran los detalles de implementación del agente inteligente descrito conceptualmente en la sub-sección 4.3.1.

En esa misma figura se puede observar la representación gráfica de un objeto de *software* denominado *agente* cuyos métodos y atributos se corresponden con cada uno de los conjuntos y funciones descritos en la 7-tupla mostrada en la sección 4.2. Las metas del agente M_a se corresponden con un atributo dicotómico que identifica si el agente debe o no optimizar la mezcla. El resto de los atributos definen los preceptos P_a del agente. El primer conjunto de métodos del objeto conforman lo que conceptualmente se plantea como la función *percibe*() que se encarga de habilitar la comunicación del agente con el resto de los componentes que componen el sistema. Los métodos que conforman la función $planifica()$ se pueden observar en la zona inferior del grupo total de métodos y conforman el núcleo de la resolución del problema de optimización. En este caso el conjunto B_a esta representado por las funciones basadas en MSV que se construyen en el método *entrena*() y se hacen utilizables a través del método *predCal*().

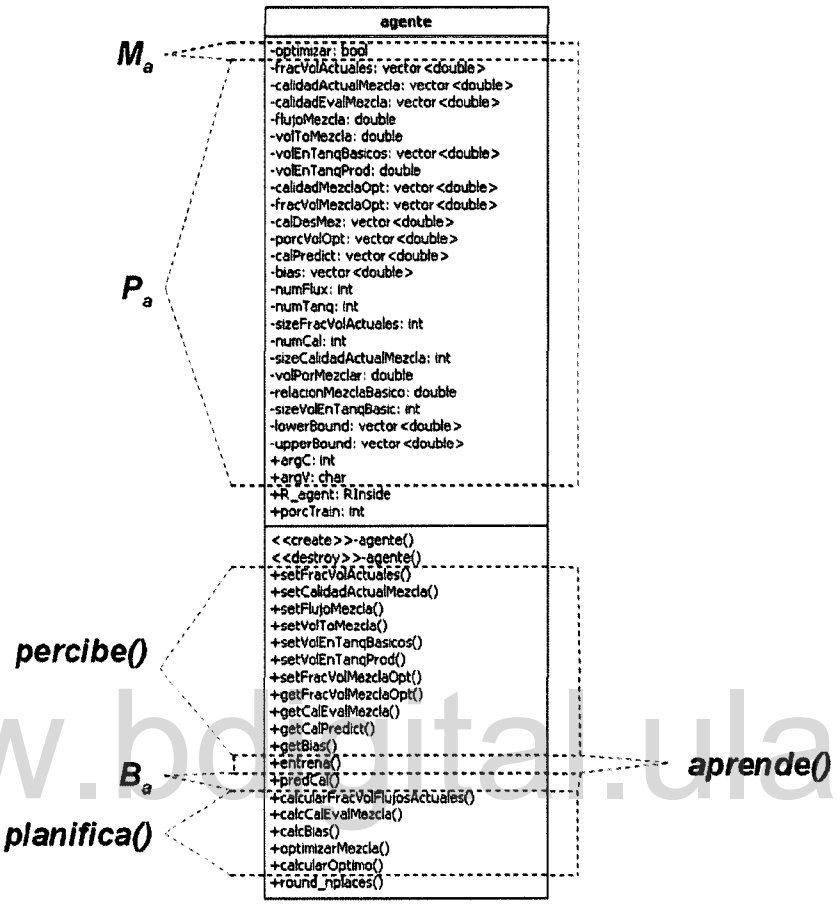

Figura 4.2: Agente inteligente como objeto de software

En la figura 4.3 se puede observar gráficamente los detalles de implementación del resto de los objetos de *software* que habilitan las funcionalidades del agente inteligente y que fueron previamente descritos en la sub-sección 3.3.2. Es importante enfatizar que el alcance de este estudio se extiende a la implementación de las clases *agente* y *My_Evaluator* ya que *RInside* y *Mads* son clases disponibles a código abierto como *software* libre.

4.4. Comentarios de implementación y validación

La sección 4.1 indica que el desarrollo de entidades de *software* concebidas como agentes inteligentes que interaccionan con un proceso industrial con el fin de optimizarlo es una estrategia que puede llevar a la obtención de grandes beneficios.

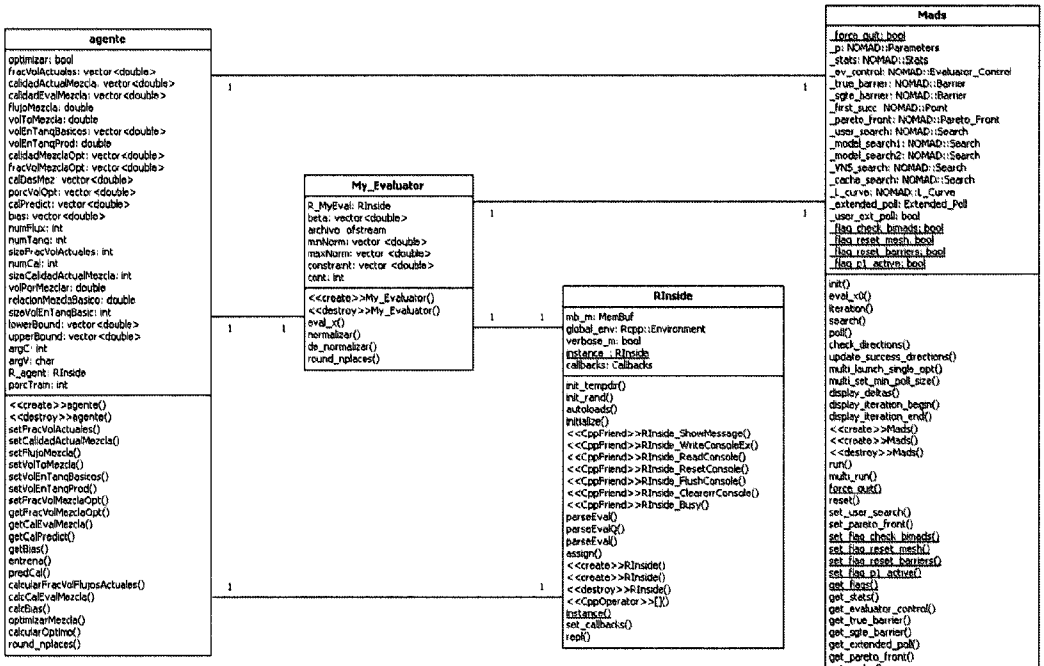

Figura 4.3: Detalles de arquitectura del agente inteligente

Entre las principales ventajas se destacan aquellas alcanzadas desde el punto de vista de arquitectura e implementación, que se evidencian al contar con un enfoque modular del diseño que permite encapsular tareas complejas como la estrategia de optimización con restricciones especiales planteada en el capítulo 3 de este trabajo.

Por otra parte, en la figura 1.4 se puede observar los diversos elementos del sistema real con quienes el agente propuesto en este trabajo debe interactuar. Nótese que la única vía de comunicación que tiene el agente se establece a través de una relación de comunicación estrecha con el sistema de control distribuido (DCS), quien ofrece al agente información del proceso e implementa en el proceso las acciones sugeridas por el agente.

Para validar el desempeño del agente, en este trabajo se propone crear un entorno simulado donde se represente tanto el DCS como el proceso. Luego el agente debe ser sometido a un conjunto de pruebas que consideren escenarios críticos en el comportamiento del proceso. Los detalles de la validación del agente se muestran en el caso de estudio descrito en el capítulo 5.

Capítulo 5

Caso de estudio

El desarrollo de una aplicación para la optimización de un proceso, tal como se mencionó en el capítulo anterior, requiere la consideración de todos los elementos que componen el sistema real de tal forma que esta aplicación sea capaz de interactuar con cada uno de los elementos que componen el sistema. Partiendo de la definición establecida en la sección 4.2 para un agente inteligente, el sistema optimizador planteado se concibe como un agente inteligente cuyas creencias corresponden a la representación del proceso de mezclas imbuído en el optimizador, aprende mediante la generalización de la base de conocimientos a través de una regresión no lineal, sus metas consisten en mantener el proceso tan económico como sea posible, reconoce diferentes estados del proceso, planifica su actuación mediante un algoritmo de optimización y percibe su entorno a través de la comunicación con el resto de los elementos que componen el sistema automatizado de mezclas.

En esta sección de plantea un caso de estudio para validar el desempeño del agente. Inicialmente se modela e implementa un entorno donde se pueda simular la comunicación del agente con cada uno de los elementos que componen el sistema automatizado de mezclas. Para ello se agregan al modelo de objetos mostrado en la Figura 4.3 dos objetos adicionales denominados *dcs* y *proceso* que representan el sistema de control distribuido y el proceso real respectivamente. El modelo del proceso se valida utilizando como referencia los indicadores estadísticos porcentaje promedio del error (PPE), raiz cuadrada el error cuadrático medio (RECM), error de validación cruzada y diagramas de paridad, comparando además con modelos basados en regresión multilineal (RML). Una vez implementado el entorno se simulan diversos escenarios operacionales donde se verifican las capacidades del agente para llevar a cabo una optimización en condiciones estáticas. Finalmente se realiza una comparación del desempeño del agente frente a un sistema basado en técnicas clásicas de optimización que utiliza RML para modelar la mezcla de propiedades.

5.1. Modelado e implementación del entorno

Se modeló un entorno que considera la interacción del agente con cada uno de los elementos del sistema tal como ocurriría en el caso real. En la Figura 5.1, se puede observar conceptualmente la arquitectura del entorno propuesto para realizar la validación del agente optimizador. Este entorno está basado en la descripción realizada en la sección 1.3.2 respecto de estrategias utilizadas para la optimización en tiempo real de mezclas.

A continuación se hace una descripción general de cada uno de los elementos que componen el entorno simulado.

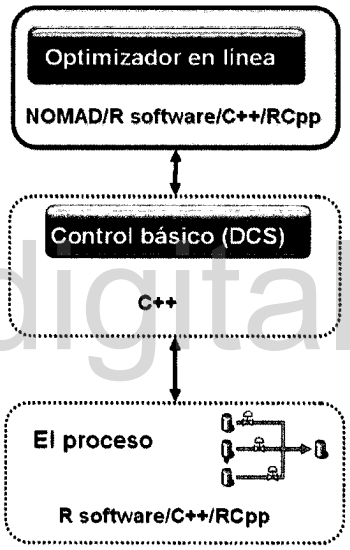

Figura 5.1: Arquitectura del sistema simulado

- El agente optimizador. Es el encargado de ejecutar los cálculos correspondientes a la optimización en tiempo real utilizando la bilioteca NOMAD bajo el lenguaje de programación C++. Las restricciones que modelan las propiedades de la mezcla son funciones basadas en máquinas de soporte vectorial construidas en el programa R.

- El simulador del proceso. Representa el proceso de mezcla como un ente que cambia de acuerdo a su interacción con los otros elementos del sistema, específicamente con el agente optimizador a través del sistema de control (DCS). Su modelado se realiza a través de funciones creadas en R.

- El sistema de control (DCS). Este elemento recibe instrucciones del agente optimizador las cuales ejecuta sobre el modelo del proceso. También es el encargado de comunicar al agente el estado del proceso.

Los modelos matemáticos para la estimación de las propiedades de la mezcla tanto en el modelo del proceso como en el agente optimizador son funciones a las cuales se accede a través de envoltorios en C++ para R. Considerando que la biblioteca NOMAD está implementada en el lenguaje C++, se utilizó este lenguaje para implementar cada uno de los elementos del entorno. La integración del software R con C++ se realiza a través de las bibliotecas RCpp (Eddelbuettel & François, 2011) y RInside (Eddelbuettel & François, 2010).

Seguidamente se describe con detalle cada uno de los elementos del entorno.

5.1.1. El simulador del proceso

Para modelar e implementar un simulador del proceso de mezclas se realizó inicialmente su modelado como un objeto de *software*, considerando la descripción del proceso y la identificación de los componentes de un sistema automatizado de mezclas. Para ello se describen los atributos y los métodos que identifican este objeto y posteriormente se realiza modelo UML. Asimismo, se consideran los métodos a través del cual este objeto logra comunicarse con el resto de los objetos que componen el sistema. El modelado del proceso como un objeto de *software* se puede observar en la Figura 5.2.

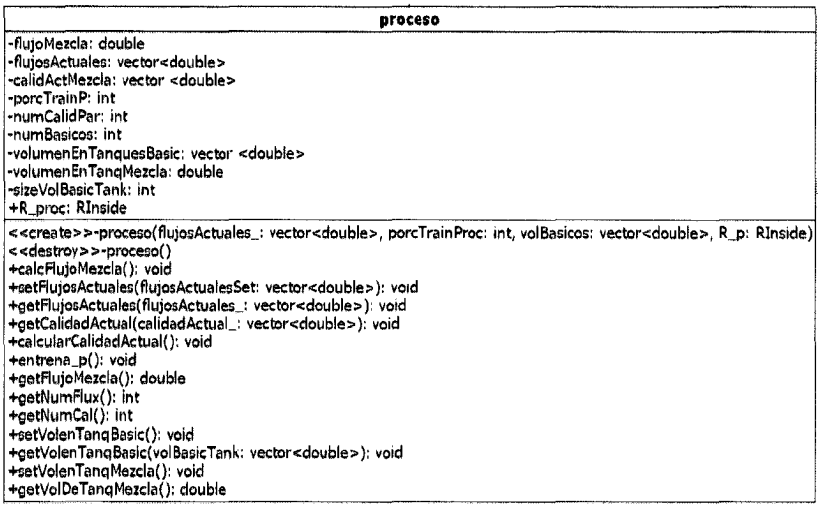

Figura 5.2: Diagrama UML del objeto *proceso*

Las funciones más importantes de este objeto son *entrena_p()* y *calcularCalidadActual()* ya que son las responsables de construir los modelos de mezcla de propiedades y predecir con ellos respectivamente. Los modelos de mezcla de propiedades se construyen utilizando MSV y R como motor de cálculo. Específicamente se utiliza la función ksvm del paquete Kernlab para el entrenamiento y la función predict para las predicciones. El vínculo entre R y C++ se realizó con las bibliotecas Rcpp (Eddelbuettel & François, 2011) y RInside (Eddelbuettel & François, 2010).

Como fuente de aprendizaje en *entrena_p()* se utiliza un conjunto de datos operacionales provenientes de la refinería de Amuay del circuito nacional de refinación de PDVSA, correspondiente a 105 mezclas orientadas a la producción de gasolina de 91 octanos (RON=91). El conjunto de datos contiene para cada mezcla la receta global utilizada, es decir, las proporciones que describen la mezcla final, y las propiedades octanaje (RON, MON), presión de vapor Reid (RVP) y destilación D-86.

En la siguiente tabla se muestra un resumen de los parámetros de los modelos generados para la predicción de las propiedades de mezcla.

Tabla 5.1: Características de los modelos de mezcla de propiedades para el objeto *proceso*

	NVS	b	***Kernel***	σ	**EVC**
RON	103	-0.4654	RBF	0,3354	0.3452
MON	101	0.2081	RBF	0,9058	0.4952
RVP	103	0.0737	RBF	0,4670	0.1165
D-86					
IBP	95	-0.3975	RBF	0,3072	2.714
T10	101	0.3823	RBF	0,2552	3.035
T50	93	0.7868	RBF	0,2337	19.194
T90	101	0.1528	RBF	0,2854	61.961
FBP	90	0.3204	RBF	0,2214	25.043

Los modelos de mezcla generados se encuentran descritos en la Tabla 5.1 en términos de número de vectores soporte (NVS) y el coeficiente de intersección (b). También se muestra el error de validación cruzada (EVC) calculado en el proceso de construcción de los modelos, el tipo de *Kernel* utilizado y el parámetro σ del *Kernel* . Es importante destacar que, tal como se explicó con detalles en la subsección 2.1.3, el

número de vectores soporte corresponde a los casos de x_i de la base de conocimientos donde se tienen coeficientes α_i, α_i^* diferentes de cero.

Si bien el EVC reportado en la Tabla 5.1 ofrece una idea de la exactitud de los modelos, la validación completa del simulador del proceso, específicamente de la función *calcularCalidadActual()*, se robusteció a través de la aplicación de la técnica de validación cruzada con tres particiones, es decir, los datos se dividen en tres partes, se toman dos de ellas para entrenar y se valida con la tercera parte. Esta operación se realiza tres veces, seleccionado en cada caso una pareja de particiones diferente para el entrenamiento. De esta forma se procura evitar una evaluación sesgada de los modelos. Como estadísticos para la evaluación de la validez de los modelos se considera la raiz cuadrada del error cuadrático medio (RECM) que es interpretada como el error estándar y el porcentaje promedio del error (PPE) que indica el nivel de desviación respecto de los valores reales en términos porcentuales.

Además, con el objeto de establecer una referencia respecto del desempeño de los modelos desarrollados, se construyen modelos de mezcla basados regresión multilineal (RML). Los modelos de mezcla de propiedades basados en RML son típicamente utilizados en las áreas operacionales para ejecutar actividades de programación de mezclas según el procedimiento operacional explicado en la sección 1.3.2. Es importante resaltar que el procedimiento de validación cruzada con tres particiones se aplicó a los modelos MSV y RML en paralelo. Los detalles respecto del modelado de mezcla de propiedades utilizando RML se muestran en el Apéndice A.

Asimismo, se genera un diagrama de paridad para las predicciones realizadas en la tercera iteración con el objeto de comparar las correlaciones entre los datos reales y los datos predichos para los dos tipos de modelo en estudio.

Los resultados de la validación cruzada para la propiedad RON en términos de los estadísticos considerados se muestran en las Figuras 5.3 a 5.5.

En la Figura 5.3 se puede observar que en las tres iteraciones correspondientes a los diferentes ensayos considerados en la validación cruzada, el error estándar (RECM) asociado a las predicciones realizadas con las funciones basadas en RML es 35.5 % mayor que el cometido por las funciones basadas en MSV. La Figura 5.4 correspondiente al PPE muestra una tendencia similar. Este comportamiento se reproduce también al estudiar el diagrama de paridad mostrado en la Figura 5.5 donde además se indica el coeficiente de correlación (R^2) y la pendiente (m) de la recta que mejor se ajusta a los puntos. Considerando que el mejor modelo será aquel cuyos valores R^2, m y corte en las absisas (b) sean lo más cercano posible a 1, 1 y 0 respectivamente; basado en la Figura 5.5 se puede concluir que en el caso del RON el modelo basado en MSV es

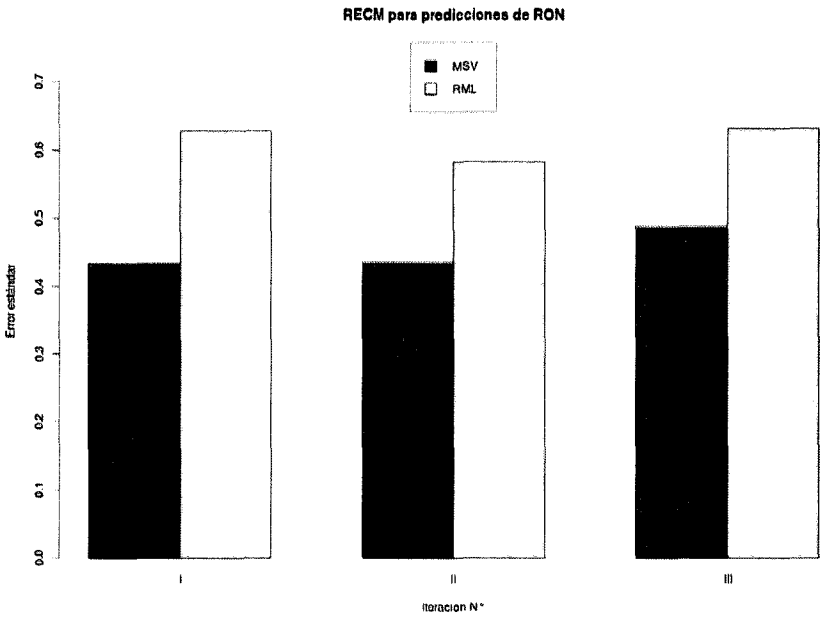

Figura 5.3: RECM para RON en la validación cruzada

más exacto que el modelo basado en RML ya que $R^2_{MSV} = 0,63 > R^2_{RML} = 0,43$, $m_{MSV} = 0,51 > m_{MSV} = 0,5$ y $b_{MSV} < b_{RML}$.

Resultados similares en incluso más contundentes en cuanto a la superioridad de los modelos basados en MSV se observan sistemáticamente al estudiar su comportamiento en la predicción del resto de las propiedades. Únicamente en el caso del T10, en la Figura 5.6, se observa que en la tercera iteración de la validación el RECM para el modelo basado en MSV es mayor al RECM del modelo basado en RML. También se observa que si bien en las otras dos iteraciones ocurre lo contrario, el desempeño del modelo basado en MSV es sólo ligeramente superior al otro modelo. El PPE para T10 mostrado en la Figura 5.7 mantiene la misma tendencia. Este indicador, en el mejor de los casos, es superior para el caso MSV solamente por un 0,2 %. Por otra parte, el diagrama de paridad de la Figura 5.8 muestra que $R^2_{MSV} = 0,32 < R^2_{RML} = 0,34$, mientras que $m_{MSV} = 0,4 > m_{MSV} = 0,5$ y $b_{MSV} < b_{RML}$. Estos indicadores demuestran que en el caso del T10 los modelos poseen un desempeño similar en cuanto al error cometido en la predicción.

Estos resultados muestran que las máquinas de soporte vectorial no sólo son una técnica conveniente para modelar propiedades de mezcla a partir de datos operacionales, sino que su desempeño es superior a las técnica comúnmente utilizada para modelar mezclas en el proceso de programación en las áreas operacionales la cual está

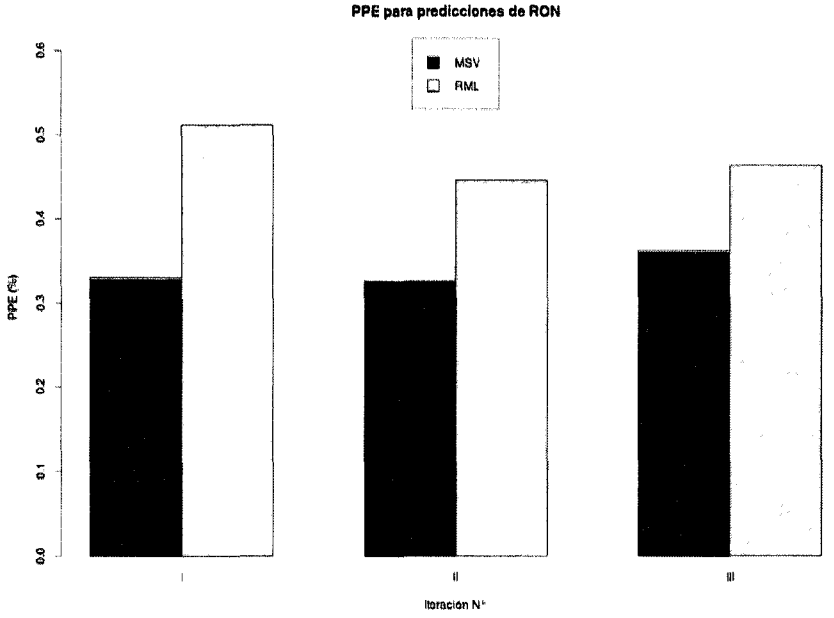

Figura 5.4: PPE para RON en la validación cruzada

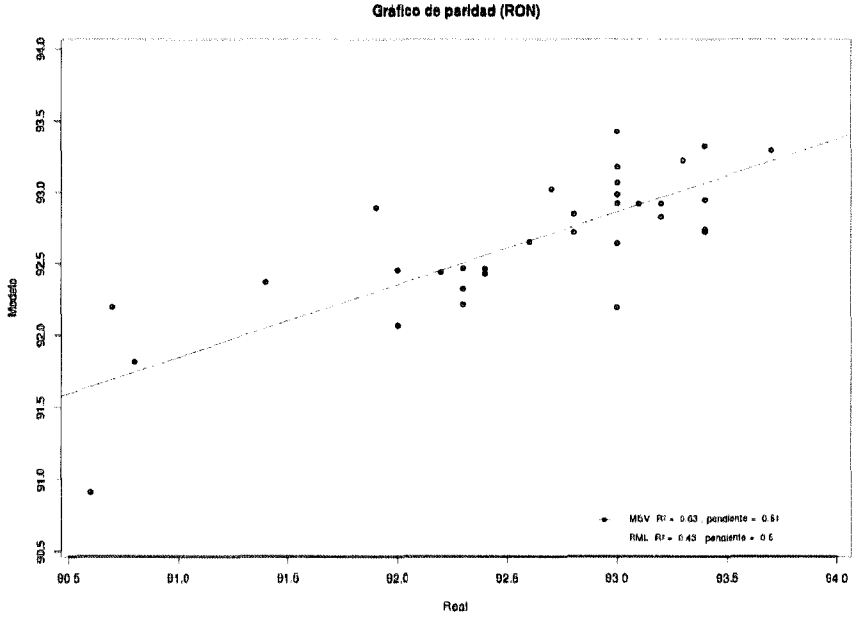

Figura 5.5: Diagrama de paridad para predicciones de RON

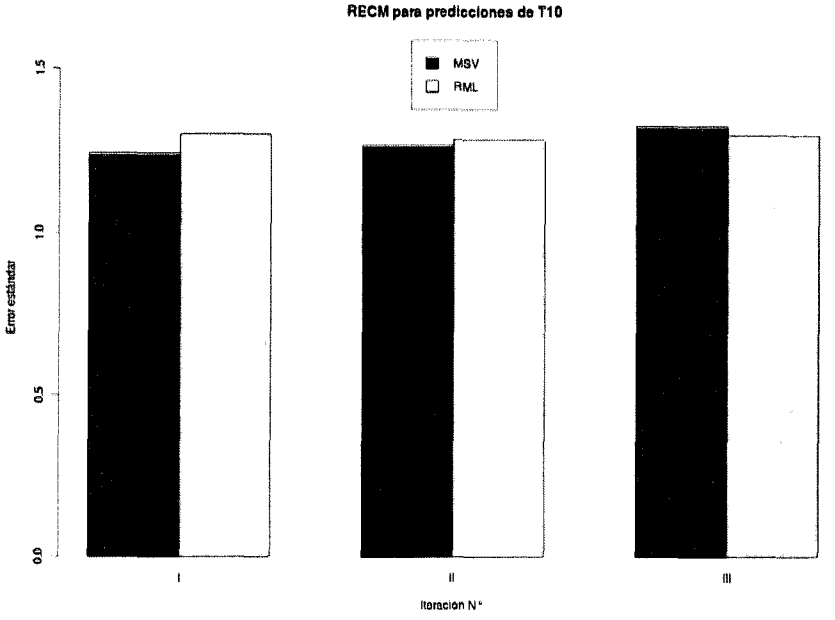

Figura 5.6: RECM para T10 en la validación cruzada

basada en una regresión lineal con ajuste no-lineal, tal como las técnicas descritas en el Capítulo 1.

El resto de los resultados de la validación cruzada en términos de los indicadores considerados, se muestran en el Apéndice B.

5.1.2. El dcs

El elemento de control básico, al igual que el resto de las entidades que componen el sistema, fue modelado como un objeto de *software* utilizando la notación estándar UML, prestando especial atención a los métodos requeridos para la comunicación con el resto de los objetos.

El modelado del elemento de control básico como un objeto de *software* se puede observar en la Figura 5.9.

Una vez modelado de control básico, se realizó la implementación de este objeto utilizando el mismo lenguaje de programación utilizado para los dos casos anteriores.

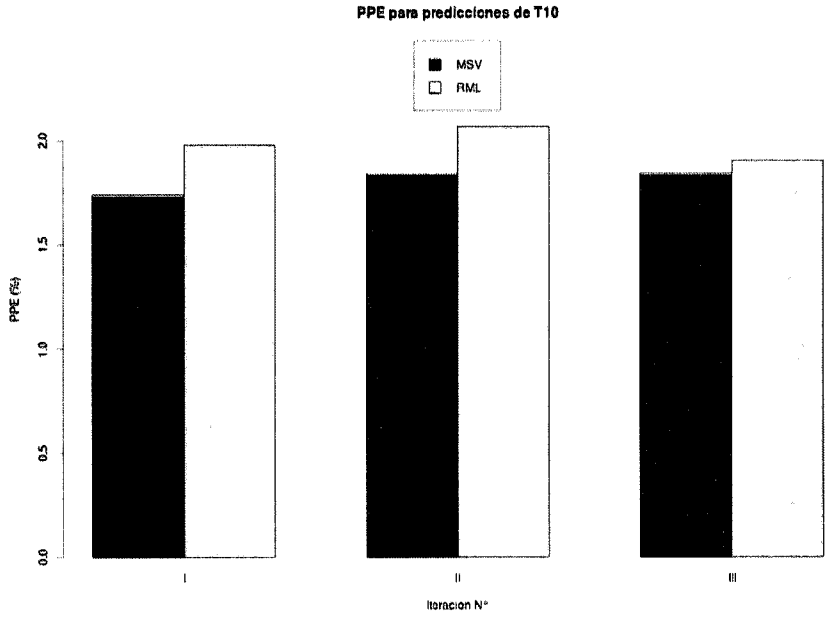

Figura 5.7: PPE para T10 en la validación cruzada

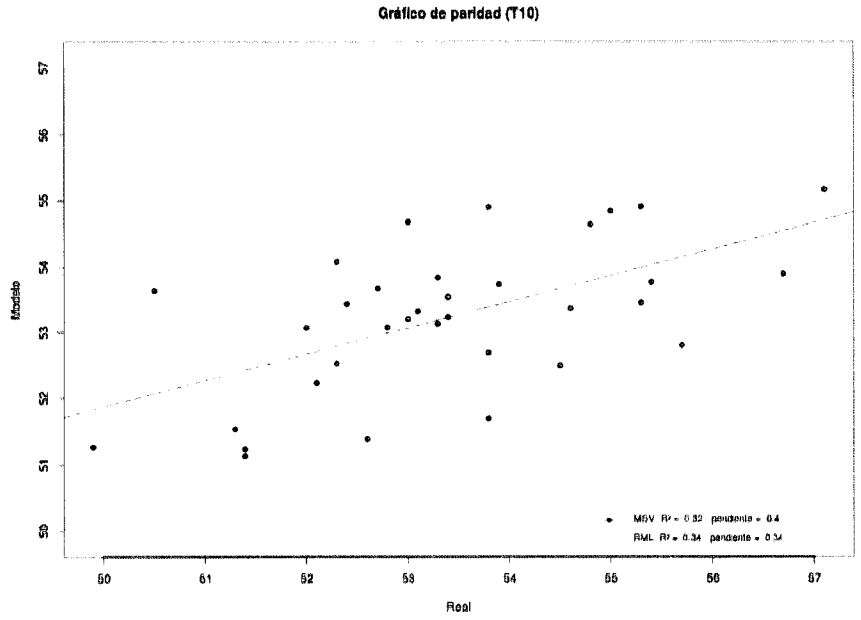

Figura 5.8: Diagrama de paridad para predicciones de T10

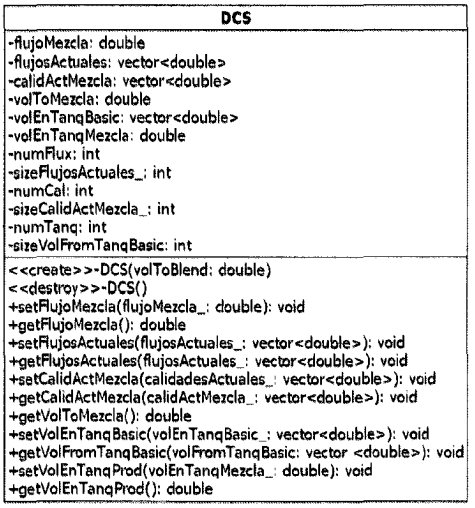

Figura 5.9: Modelado del elemento de control básico como un objeto de software

5.2. Configuración del agente

Para este caso de estudio el agente descrito en la sección 4.3 se implementa validar su desempeño en condiciones estáticas atendiendo a la configuración descrita a continuación.

El conjunto de creencias B_a esta representado por las funciones basadas en MSV descritas en la sección 2.3 que se construyen en el método $entrena()$ y se hacen utilizables a través del método $predCal()$. Es importante destacar que para el entrenamiento del agente no se utilizó la totalidad de los datos operacionales disponibles como fuente de aprendizaje, utilizándose sólo las dos terceras partes de ellos. De esta forma se prueba la capacidad de generalización del agente.

Las metas del agente M_a se corresponden con un atributo dicotómico que identifica si el agente debe o no optimizar la mezcla. En este caso, donde el objetivo es validar el desempeño en condiciones estáticas, este atributo siempre es positivo indicando la necesidad de llevar a cabo el proceso de optimización.

La función $planifica()$ está basada en la estrategia de optimización descrita en la sección 3.3 que propone la utilización del algoritmo MADS a través de su implementación en *software* libre NOMAD.

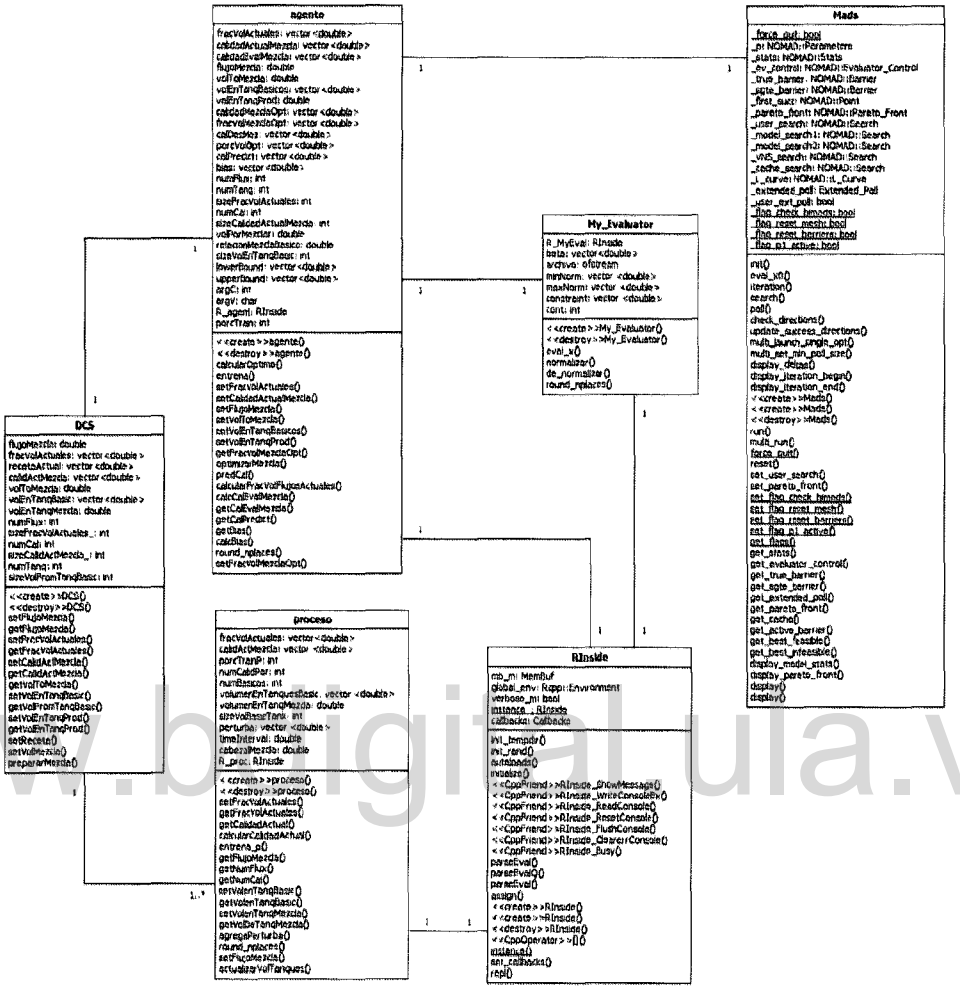

Figura 5.10: Modelado integrado del sistema simulado de mezclas

5.3. Modelado del sistema integrado

El modelado del sistema integrado agente/entorno se realizó a través de un diagrama de objetos, un diagrama de actividades y un diagrama de secuencia.

Referencialmente el diagrama de objetos se puede observar en la Figura 5.10; sin embargo, se puede ver con detalles en el Apéndice C. En este diagrama de objetos se muestra la interacción entre cada uno de los componentes del sistema, por lo que a través de su inspección se puede garantizar la consideración de todos los canales de comunicación necesarios entre el agente y el entorno simulado.

El diagrama de actividades donde se muestra las acciones de cada uno de los objetos en un ciclo de optimización se muestra referencialmente en la Figura 5.11; sin

embargo, se puede ver con detalles en el Apéndice D.

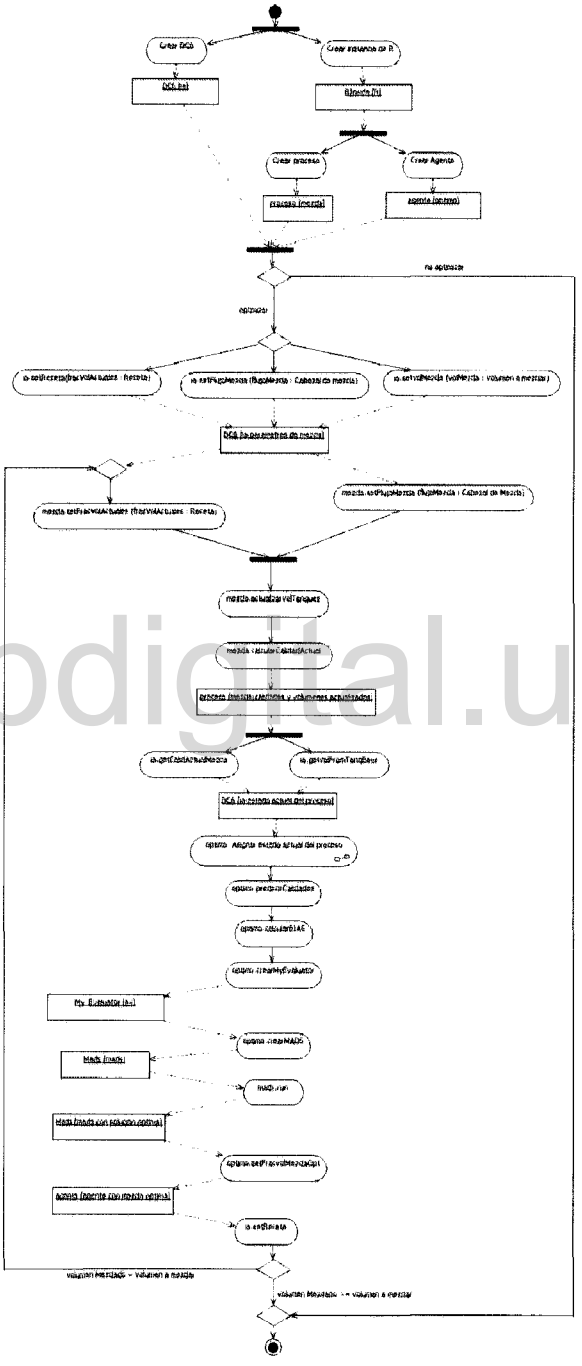

Figura 5.11: Diagrama de actividades de la interacción agente/entorno

El diagrama de secuencia se muestra referencialmente en la Figura 5.12, pero se

muestra con mayor claridad en el Apéndice E.

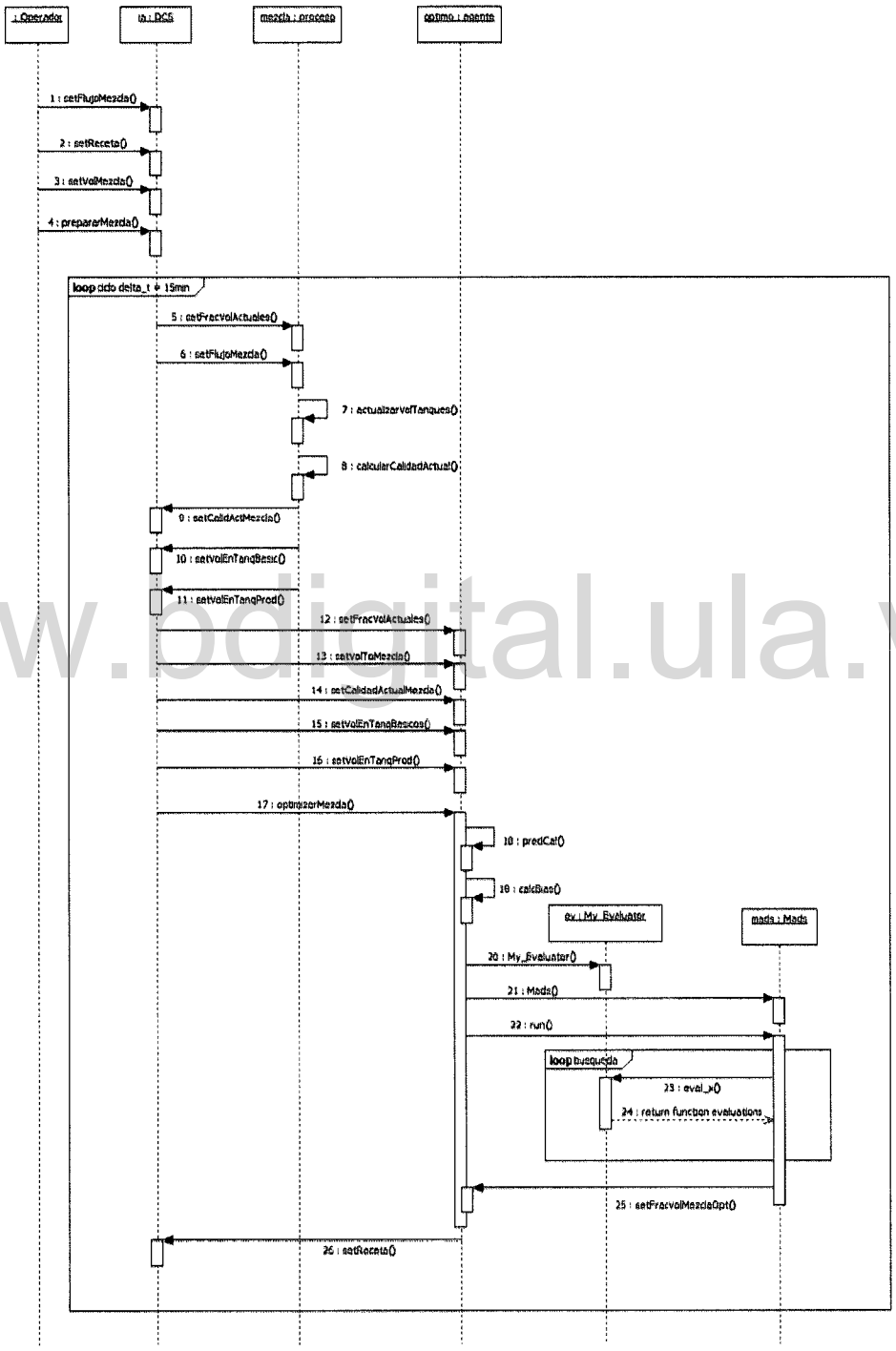

Figura 5.12: Diagrama de secuencia del sistema integrado agente/entorno

Una vez modelado el sistema integrado su implementación se realizó utilizando el lenguaje de programación C++. Esta etapa consiste principlamente en la codificación de de la interacción de cada uno de los elementos u objetos previamente implementados, de acuerdo a la lógica establecida en los diagramas elaborados.

5.4. Evaluación del desempeño del agente

El sistema integrado por el simulador de mezcla, el elemento de control básico y el agente optimizador, posteriormente, fue utilizado para simular casos reales donde se pone a prueba la capacidad del agente optimizador, con el objeto de determinar su eficacia para resolver el problema de optimización y eficiencia en tiempo de ejecución. Con este objetivo inicialmente se realizó un ejercicio de optimización estática, es decir, el primer paso del ciclo mostrado en los diagramas de las Figuras 5.11 y 5.12 para optimizar un proceso de mezcla desde su inicio hasta su finalización. Para ello se utiliza como estimados iniciales cada una de las observaciones previamente realizadas por el agente en el entrenamiento y que conforman su base de conocimientos. Posteriormente, en el segundo ciclo de optimización, con el objeto de crear una mayor diferencia "modelo-proceso", se agregaron ciertas "perturbaciones" en forma de desviaciones constantes añadidas a las propiedades calculadas por el simulador de mezcla. Este sería el comportamiento observado en el sistema real ante cambios inesperados en las propiedades de los básicos. Los resultados obtenidos tanto para el primer ciclo de optimización (caso estático) como al segundo ciclo de optimización (caso dinámico) se muestran en las próximas subsecciones.

5.4.1. Optimización en condiciones estáticas

Las condiciones bajo las cuales fue realizada la optimización estática se muestran en la Tablas 5.2, 5.3 y 5.4.

Tabla 5.2: Disponibilidad inicial de básicos en inventario (BBL) en la optimización inicial

LVN	LKN	HCCN	ALQ	nC4	MTBE	REF	TAME
22000	18000	110000	120000	20000	130000	20000	10000

En la Tabla 5.2, se muestra la disponibilidad inicial de básicos en el inventario. Este inventario posee gran importancia ya que representa el límite superior en la disponibilidad de algún básico para la mezcla formando parte así de las restricciones del

proceso de optimización. Adicionalmente, en las Tablas 5.3 y 5.4 se muestra la receta inicial considerada para el proceso de mezcla, las calidades simuladas obtenidas del proceso y la estimación realizada por el agente optimizador. Estos datos representan los estados del proceso y del agente justo después del método 19:calcBias() del diagrama mostrado en la Figura 5.12. Esto indica que el proceso acaba de iniciar, el agente observa y tiene una percepción del mismo. Es importante destacar que si bien en la Tabla 5.4 se muestra que la percepción del agente antes de ejecutar el primer ciclo de optimización es que el proceso esta en especificación, de igual forma el agente procura conseguir una mejor mezcla en términos económicos a través de la función $planifica()$.

Tabla 5.3: Receta(%) antes del primer ciclo de optimización del agente

LVN	LKN	HCCN	ALQ	nC4	MTBE	REF	TAME	($/BBL)
15	5	0	20	0	50	10	0	63,72

Tabla 5.4: Especificaciones de mezcla y predicción de calidades antes del primer ciclo de optimización del agente

Especificación	Sim. Proceso	Estim. Agente	BIAS
RON $\geqq$ 91,6	92,2	92,3	+0,1
RVP$\leqq$ 9,5	8,85	8,91	+0,06
RVP$\geqq$ 8,9	8,85	8,91	−0,06
IAD $\leqq$87,6	87,4	87,5	+0,1
IAD $\geqq$ 87	87,4	87,5	−0,1
Destilación			
IBP $\geqq$30	34,0	34,0	0
T10 $\leqq$ 70	54,1	53,6	-0,5
T50 $\geqq$77	88,6	88,4	-0,2
T50 $\leqq$ 121	88,6	88,4	-0,2
T90 $\leqq$195	168,2	168,5	+0,3
FBP $\leqq$225	208,5	207,9	-0,6

Los resultados obtenidos de esta optimización inicial se muestran en las Tablas 5.5 y 5.6, mientras que el reporte completo de este primer ejercicio de optimización se muestra en el Apéndice F, específicamente en el Apéndice F.1.

Tabla 5.5: Receta obtenida del primer ciclo de optimización del agente

LVN	LKN	HCCN	ALQ	nC4	MTBE	REF	TAME	($/BBL)
20,4	18,1	44,0	1,2	4,4	9,1	2,0	0,8	52,3

Tabla 5.6: Especificaciones de mezcla y predicción de calidades después del primer ciclo de optimización del agente

Especificación	Sim. Proceso	Estim. Agente	BIAS
RON $\geqq$ 91,6	92,2	92,2	0
RVP$\leqq$ 9,5	9,02	8,96	-0,08
RVP$\geqq$ 8,9	9,02	8,96	-0,08
IAD $\leqq$87,6	87,5	87,5	0
IAD $\geqq$ 87	87,5	87,5	0
Destilación			
IBP $\geqq$30	34,0	33,6	-0,4
T10 $\leqq$ 70	53,1	54,0	−0,9
T50 $\geqq$77	88,2	88,0	-0,2
T50 $\leqq$ 121	88,2	88,0	-0,2
T90 $\leqq$195	168,7	168,2	-0,5
FBP $\leqq$225	208,0	208,1	+0,1

En la Tabla 5.4 se puede observar que la receta inicial mostrada en la Tabla 5.3 viola la especificación RVP$\geqslant$8,9. Es decir, de continuar la mezcla con esta receta se obtendría un producto completamente fuera de especificación para el RVP. Una vez llevada a cabo la intervención del agente a través de la realización de una optimización del proceso, la receta obtenida se muestra en la Tabla 5.5. Las calidades resultantes en el producto, según los datos observados en la Tabla 5.6, indican que la optimización logró un ajuste en los parámetros de calidad de la gasolina producida. Asimismo, de la Tabla 5.5 se puede observar que hubo una disminución sustancial del costo de producción de 63,72 $/Barril a 52,3 $/Barril.

En la Figura 5.13 se muestra la evolución del costo de producción de un barril de combustible respecto del número de iteración del algoritmo MADS. Nótese que el algoritmo genera un comportamiento rápidamente convergente hacia una solución que se ubica entre las opciones menos costosas de las evaluadas. Además, considerando

que un ciclo de optimización en línea de mezclas en sistemas comerciales oscila entre los 10 y 15 min, el tiempo de respuesta del algoritmo (98 s) se considera adecuado.

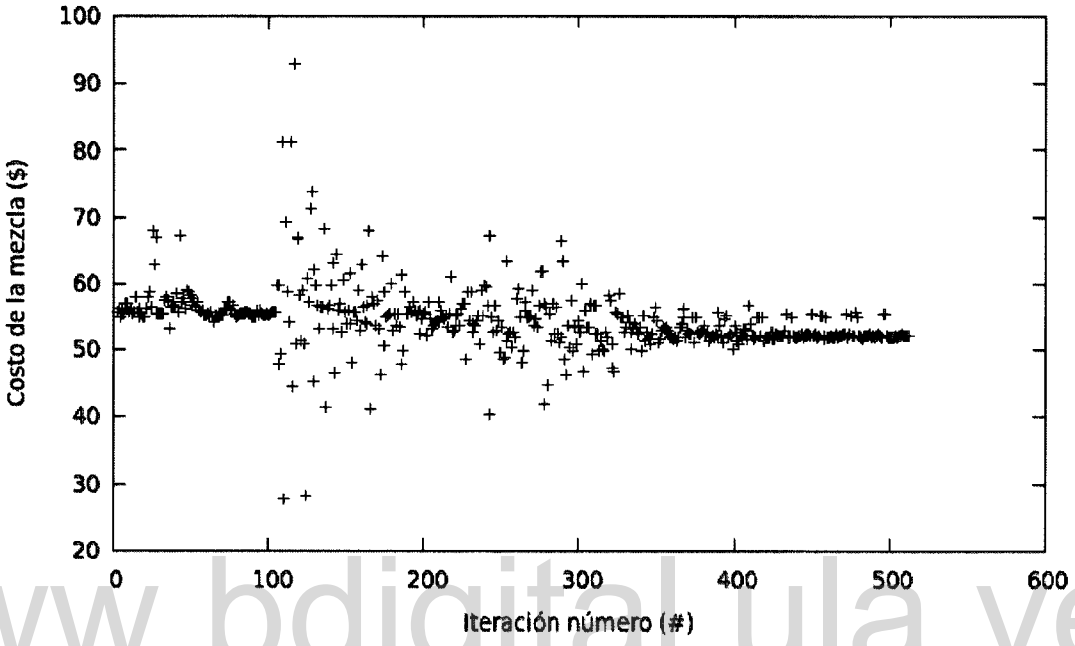

Figura 5.13: Costo de la mezcla vs. número de iteración en el primer ciclo de optimización

5.4.2. Optimización con perturbaciones

Una vez comprobada la eficacia de la estrategia propuesta para realizar el primer ciclo de la optimización, se procedió simular el segundo ciclo donde además se incorporan "perturbaciones" en forma de desviaciones constantes en las propiedades del produto en la simulación del proceso. Estas perturbaciones representan los posibles comportamientos inesperados de las propiedades del producto obtenido durante el proceso real debido a estratificación de tanques o perturbaciones en los procesos aguas arriba. Como orden de magnitud de las perturbaciones se consideró las fluctuaciones típicas encontradas en los históricos del proceso para la producción de básicos y su efecto en la mezcla. Este escenario representa una de las principales causas de desviación en las especificaciones de productos finales.

Las perturbaciones agregadas se muestran en la Tabla 5.7.

Tabla 5.7: Perturbaciones agregadas en las calidades del producto

RON	MON	RVP	PIE	T10	T50	T90	FBP
+0,1	−0,1	−0,5	-5	0	0	0	0

Si bien estas perturbaciones no alteran el costo final del producto generado, sí modifican las calidades del mismo hasta el punto en el que sus propiedades están fuera de las especificaciones requeridas. En la Tabla 5.8, en la columna "Sim. Proceso", se muestran las especificaciones del producto de acuerdo a la simulación del proceso, previo a la optimización bajo perturbaciones. Nótese que la propiedad IBP se encuentra fuera de especificación.

Tabla 5.8: Especificaciones de mezcla y predicción de calidades antes de la optimización con perturbaciones en las calidades del producto

Especificación	Sim. Proceso	Estim. Agente	BIAS
RON ≧ 91,6	92,3	92,3	0
RVP≦ 9,5	9,4	8,9	-0,5
RVP≧ 8,9	9,4	8,9	-0,5
IAD ≦87,6	87,5	87,6	+0,1
IAD ≧ 87	87,5	87,6	+0,1
Destilación			
IBP ≧30	29,1	34,1	−5,0
T10 ≦ 70	53,5	53,4	-0,1
T50 ≧77	88,3	88,9	+0,5
T50 ≦ 121	88,3	88,9	−0,5
T90 ≦195	168,6	169,1	+0,5
FBP ≦225	208,3	207,9	-0,4

En la Tabla 5.9, se muestra la receta obtenida como resultado de la optimización en un proceso perturbado. Además, en la Tabla 5.10 se muestran las especificaciones de calidad del producto una vez realizada la optimización. En el Apéndice F, específicamente en el Apéndice F.2, se muestra el reporte completo de este segundo ejercicio de optimización.

Es importante destacar que aunque hubo un incremento en el costo del barril de combustible producido (53,16 $/BBL), lo más importante de este segundo ciclo de

optimización es mantener el producto generado dentro de especificaciones de calidad para evitar una corrección al final de la mezcla lo cual sí acarrearía un enorme incremento del costo de producción.

Tabla 5.9: Receta obtenida luego de la optimización con perturbaciones en las calidades del producto

LVN	LKN	HCCN	ALQ	nC4	MTBE	REF	TAME	($/BBL)
19,3	16,5	47,7	0,5	1,0	13,4	0,5	0,7	53,16

Tabla 5.10: Especificaciones de mezcla y predicción de calidades luego de la optimización con perturbaciones en las calidades del producto

Especificación	Sim. Proceso	Estim. Agente	BIAS
RON $\geqq$ 91,6	92,6	92,4	-0,2
RVP$\leqq$ 9,5	9,5	9,4	-0,1
RVP$\geqq$ 8,9	9,5	9,4	-0,1
IAD $\leqq$87,6	87,3	87,1	-0,2
IAD $\geqq$ 87	87,3	87,1	-0,2
Destilación			
IBP $\geqq$30	30,0	30,0	0
T10 $\leqq$ 70	52,5	52,8	+0,3
T50 $\geqq$77	80,7	80,9	+0,2
T50 $\leqq$ 121	80,7	80,9	+0,2
T90 $\leqq$195	164,0	163,2	-0,8
FBP $\leqq$225	203,9	204,1	+0,2

En la Figura 5.14 se muestra la evolución del costo de producción de un barril de combustible respecto del número de iteración del algoritmo MADS para este segundo ejercicio de optimización. Nótese que el tiempo de respuesta del algoritmo se sigue considerando adecuado ya que el resultado fue alcanzado en un tiempo de 111 s.

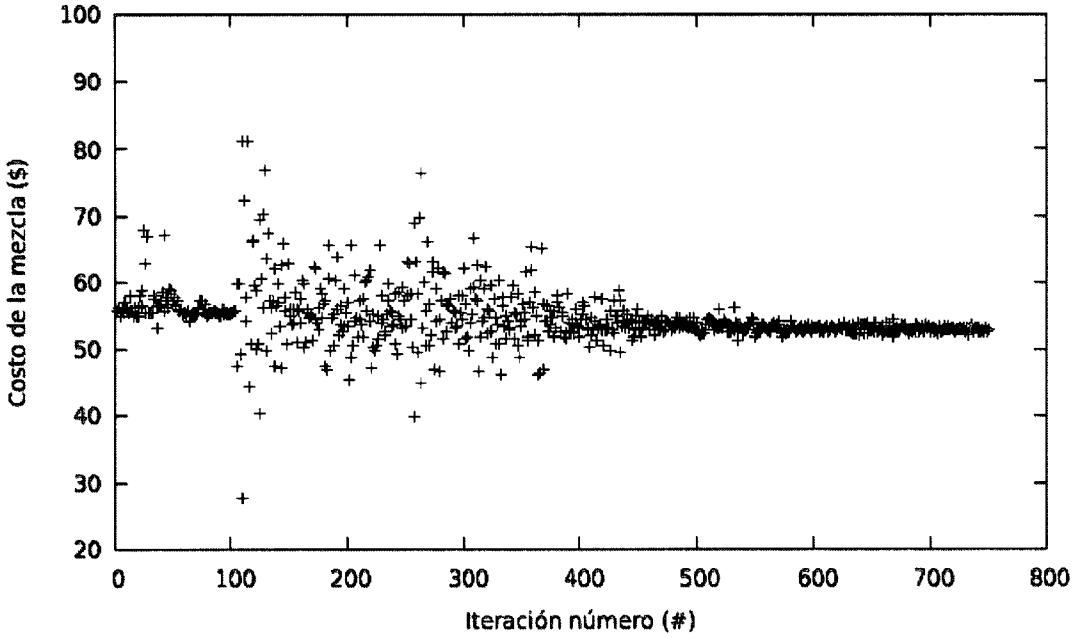

Figura 5.14: Costo de la mezcla vs. número de iteración en la optimización con perturbaciones

5.5. Comparación del desempeño del agente contra una estrategia clásica de optimización

El desempeño del agente desarrollado fue comparado con los resultados arrojados por una estrategia de optimización basada en el método de los multiplicadores de Lagrange aumentado con un algoritmo de programación cuadrática sucesiva (SQP) interior, de una forma similar a la implementada por la aplicación comercial Lancelot. Específicamente se utilizó el paquete RSolnp (Ghalanos & Theussl, 2012) disponible como *software* libre para la aplicación R.

Se estudiaron dos casos. El primero de ellos plantea un escenario donde existen ligeras limitaciones en la disponibilidad de todos los básicos. En el segundo de ellos se muestra un escenario donde existe total disponibilidad de algunos de ellos, pero severa indisponibilidad de otros.

5.5.1. Escenario A

El escenario A se describe con detalles en la Tabla 5.11. Nótese que para una mezcla de 100 000 BBL existen disponibilidades de los básicos a lo sumo de 50 %.

LVN	LKN	HCCN	ALQ	nC4	MTBE	REF	TAME
25000	25000	50000	15000	12000	30000	20000	20000

Tabla 5.11: Escenario A. Limitaciones en la disponibilidad de todos los básicos (BBL)

Los resultados de la optimización de este escenario tanto para el agente como para la estrategia basada en Rsolnp se muestran en las Tablas 5.12 y 5.13. En la Tabla 5.12 se puede observar que ambas estrategias alcanzan soluciones factibles de acuerdo a los modelos que definen sus restricciones. También son soluciones factibles de acuerdo a la predicción del simulador de proceso (Sim. proc) para cada una de las recetas, tanto la propuesta por el agente como la propuesta por RSolnp, mostradas en la Tabla 5.13. Sin embargo, la Tabla 5.13 también muestra que la solución hallada por el agente posee un menor costo. El tiempo de optimización se observó menor para la estrategia basada en RSolnp la cual propone una solución a los 12 s mientras que el agente demora 85 s.

Tabla 5.12: Especificaciones de mezcla y predicción de calidades en el escenario A

	Receta Agente		Receta RSolnp	
Especificación	**Sim. Proc.**	**Agente**	**Sim. Proc.**	**RSolnp**
RON $\geqq$ 91,6	92,2	92,3	92,3	92,6
RVP$\leqq$ 9,5	8,91	8,9	8,96	9,5
RVP$\geqq$ 8,9	8,91	8,9	8,96	9,5
IAD $\leqq$87,6	87,4	87,4	87,5	87,2
IAD $\geqq$ 87	87,4	87,4	87,5	87,2
Destilación				
IBP $\geqq$30	33,2	33,2	33,9	30,1
T10 $\leqq$ 70	53,6	53,7	53,1	51,7
T50 $\geqq$77	87,5	87,7	87,5	85,5
T50 $\leqq$ 121	87,5	87,7	87,5	85,5
T90 $\leqq$195	167,6	167,8	167,4	172,0
FBP $\leqq$225	208,5	208,5	208,5	209,4

Tabla 5.13: Recetas obtenidas y su costo en el escenario A

	LVN	LKN	HCCN	ALQ	nC4	MTBE	REF	TAME	($/BBL)
Agente	24,1	23,8	29,7	0	2,5	2,6	0,1	16,4	51,6
Rsolnp	25,0	23,8	23,2	0	1,9	3,5	4,0	18,6	52,8

5.5.2. Escenario B

El escenario B se describe con detalles en la Tabla 5.14. Nótese que para una mezcla de 100 000 BBL existe total disponibilidad para el HCCN, nC4, MTBE, REF; mientras que para el LVN, el LKN y el TAME las disponibilidades son sólo de 1 %, 10 % y 1 % respectivamente.

LVN	LKN	HCCN	ALQ	nC4	MTBE	REF	TAME
1000	10000	100000	1000	100000	100000	100000	1000

Tabla 5.14: Escenario B. Limitaciones severas en la disponibilidad de algunos básicos (BBL)

Los resultados de la optimización de este escenario tanto para el agente como para la estrategia basada en Rsolnp se muestran en las Tablas 5.15 y 5.16. En la Tabla 5.15 se puede observar que la solución hallada por el agente posee un costo mayor que la solución encontrada con Rsolnp. Sin embargo, en la Tabla 5.16 se muestra que la solución hallada por la estrategia basada en RSolnp no es factible al considerar la predicción del simulador de procesos al simular las calidades de los productos para dicha receta. Específicamente se puede observar que la restricción (RVP $\geqq$ 8,9) no se satisface según la predicción del simulador del proceso para la receta propuesta por RSolnp. En este caso el tiempo de optimización tambień se observó menor para la estrategia basada en RSolnp la cual propone una solución a los 10 s mientras que el agente demora 87 s.

Tabla 5.15: Recetas obtenidas y su costo en el escenario B

	LVN	LKN	HCCN	ALQ	nC4	MTBE	REF	TAME	($/BBL)
Agente	0,9	1,1	1,4	0,8	10,9	23,8	60,5	0,6	66,2
Rsolnp	1,0	10,0	69,8	0	3,3	15,9	0	0	55,3

Tabla 5.16: Especificaciones de mezcla y predicción de calidades en el escenario B

	Receta Agente		Receta RSolnp	
Especificación	**Sim. Proc.**	**Agente**	**Sim. Proc.**	**RSolnp**
RON $\geqq$ 91,6	92,1	92,1	92,4	93,1
RVP$\leqq$ 9,5	9,00	8,95	8,83	8,90
RVP$\geqq$ 8,9	9,00	8,95	8,83	8,90
IAD $\leqq$87,6	87,2	87,3	87,5	87,6
IAD $\geqq$ 87	87,2	87,3	87,5	87,6
Destilación				
IBP $\geqq$30	34,7	34,8	34,0	33,8
T10 $\leqq$ 70	55,6	55,7	53,8	52,0
T50 $\geqq$77	95,3	90,8	89,1	80,4
T50 $\leqq$ 121	95,3	90,8	89,1	80,4
T90 $\leqq$195	164,4	168,7	169,5	167,4
FBP $\leqq$225	207,3	207,3	208,9	206,6

5.6. Aspectos económicos

La ejecución de una estrategia de optimización en línea de mezclas genera diversos beneficios para las operaciones de manejo de productos terminados en una refinería. El primero de ellos es la minimización de regalos de calidad lo cual se evidencia en la disminución de los costos de producción del combustible. Por otra parte, la eliminación de las re-mezclas es un aspecto de gran importancia ya que esta operación incrementa el costo del combustible producido al utilizar básicos de alto valor agregado, y por lo tanto altos costos, para lograr la corrección. Asimismo, la eliminación de la re-mezcla incrementa la flexibilidad operacional ya que al disminuir el tiempo de utilización de un tanque de almacenamiento se evita el pago de multas por tiempo de espera de buques en puerto y se disminuye el riesgo de disminuir la producción por alto inventario.

Para realizar un estimado económico de las ventajas de implementación de la estrategia de optimización en línea de mezclas planteada en este trabajo, se tomará en cuenta el ahorro obtenido por eliminación de la re-mezcla y la consecuente multa por demoras en la descarga de combustible al buque.

Si la perturbación indicada en la Tabla 5.7 y el consecuente efecto en las calidades

de producto mostrado en la Tabla 5.8, se prolonga a lo largo de una mezcla de 70 000 Barriles, evidentemente se obtendrá una mezcla completamente fuera de especificación que requiere de una corrección. Para lograr la corrección de esta mezcla, según cálculos efectuados en el área operacional a partir de modelos propietarios, es necesaria la incorporación de 5 000 Barriles adicionales de básicos distribuidos de la siguiente forma: 305 Barriles de alquilato (ALQ), 3000 Barriles de reformado (REF) y 1695 Barriles de tame (TAME). La mezcla final obtenida al realizar esta corrección posee un costo de 55,81 $/Barril. Si se tiene en cuenta que el costo de producción del combustible al realizar el proceso de optimización con perturbaciones es de 53,16 $/Barril, el ahorro obtenido al ejecutar dicho proceso es de 2,65 $/BBL. Para una mezcla de 70 000 Barriles el ahorro sería de 185 500 $/mezcla.

Por otra parte, un proceso de corrección de mezcla se toma aproximadamente 8 horas, tomando en cuenta la dosificación de los básicos al tanque, y el tiempo de recirculación para lograr homogeneidad en el producto. Considerando que la multa por demora en la descarga del producto a buque es de 1 200 $/hora, el ahorro obtenido ascendería a la suma de 195100 $/mezcla. Si se asume que la obtención de tanques fuera de especificación en una refinería que no implemente este tipo de estrategias es el 10 % de los tanques mezclados, y que se mezcle una cantidad de 300 tanques al año, el estimado de ahorro anual estaría en el orden de los 5MM $ /año.

Capítulo 6

Conclusiones

La optimización en tiempo real es un enfoque de control de procesos basado en modelos, que utiliza información actual del proceso para predecir políticas de operación óptima para una unidad de proceso durante el próximo intérvalo de operación. En este sentido la optimización en tiempo real (OTR) del proceso de mezclas representa un oportunidad clave para la minimización de costos en el proceso de refinación.

La OTR ha sido planteada por diversos autores (Forbes *et al.* , 2002; Diaz & Barsamian, 1996; Monder, 2001; Wang *et al.* , 2007; Singh *et al.* , 1999; Sullivan, 1990) para minimizar costos o evitar la obtención de productos fuera de especificación en el proceso de mezclas de gasolina. Sin embargo, en el planteamiento del problema de optimización los modelos que representan la mezcla de propiedades generalmente son aproximaciones lineales o lineales con ajustes no-lineales, que no representan el proceso de manera exacta. Estas suposiciones frecuentemente conllevan a la obtención de una mezcla fuera de las especificaciones esperadas generando altos costos adicionales.

Las máquinas de soporte vectorial (MSV) presentadas por Vapnik y colaboradores, son un método muy poderoso que en pocos años desde su introducción ya ha superado a la mayoría de los otros sistemas de aprendizaje en un amplio campo de aplicaciones.

Con el objeto de contar con un optimizador de mezclas de gasolina más efectivo, en esta tesis se propuso obtener modelos de mezcla de propiedades basados MSV a partir de información histórica del proceso, e incorporarlos como representación de la base conocimientos de un optimizador de mezclas concebido como un agente inteligente. A través de esta estrategia se obtuvieron modelos de mezcla con menores diferencias modelo - proceso, lo cual fue mostrado utilizando validación cruzada. Específicamente se obtuvo que las máquinas de soporte vectorial son capaces de modelar propiedades de mezcla de gasolinas con errores de predicción sistemáticamente menores al compararlos con modelos basados en regresión multilineal considerando los estadísticos

error cuadrático medio y porcentaje promedio del error, así como la pendiente y el coeficiente de correlación del gráfico de paridad.

Los optimizadores en tiempo real clásicos normalmente utilizan aplicaciones comerciales como GAMS (General Algebraic Modeling Systems), MINOS o Lancelot que implementan variaciones de métodos de programación lineal como programación lineal sucesiva (SLP, por sus siglas en inglés), programación cuadrática sucesiva (SQP, por sus siglas en inglés) ó programación de gradiente reducido generalizado (GRG, por sus siglas en inglés). En este trabajo, considerando la complejidad algebraica para encontrar las derivadas de algunas funciones basadas en MSV, se plantea un enfoque de optimización libre de derivadas donde no es necesario el tratamiento analítico de las restricciones del problema de optimización. Para ello se utilizó NOMAD, un *software* que implementa el algoritmo MADS *(Mesh Adaptive Direct Search)* para la resolución de problemas de optimización evitando el uso de información referente a las derivadas de las restricciones. Fue así como la estrategia para el modelado de mezclas de gasolina basada en máquinas de soporte vectorial se implementó exitosamente como representación de la base de conocimientos de un dispositivo para la optimización en tiempo real de mezclas concebido como un agente inteligente.

Para la validación del agente inteligente, se creó un entorno simulado donde el agente optimizador interactúa con el resto de los elementos del sistema, esto es, el proceso y el sistema de control distribuido (DCS). Allí se comprobó su eficacia tanto para la obtención del óptimos en condiciones estáticas, es decir, en la primera iteración del ciclo de optimización, como para manejar perturbaciones en las calidades del producto en el segundo ciclo de optimización (condiciones dinámicas). Allí mismo se pudo constatar que los tiempos de respuesta del agente inteligente son satisfactorios de acuerdo a la dinámica del proceso.

También se comparó para casos operacionales críticos el desempeño del agente respecto de los resultados obtenido con el paquete RSolnp que implementa técnicas determinísticas de optimización y restricciones basadas en RML. En esta oportunidad se encontró que a pesar de que el agente demora un mayor tiempo en proponer una solución, ofrece mejores resultados bien sea en la determinación de una receta más económica o en el hallazgo de recetas factibles donde RSolnp sugiere recetas infactibles según predicción del simulador de procesos para la receta propuesta como óptimo factible. Estos resultados permiten concluir que con el agente inteligente se obtienen mejores resultados en el cálculo de las recetas que deben ser indicadas al nivel de control en cada uno de los pasos de la optimización.

Por último, al realizar un análisis económico de la solución planteada por el agente en el caso de optimización en condiciones estáticas con perturbaciones, se evidenció que la implementación de esta estrategia de optimización de mezclas puede generar ahorros de hasta 5MM$ al año en una refinería de mediana capacidad del circuito de refinación de PDVSA. El escenario considerado al realizar esta estimación fue la obtención de una mezcla fuera de especificación y el pago de una multa por demora en la descarga de buques. Existen otros escenarios como la disminución de la producción por alto inventario en almacenaje que puede ser parcialmente solventado al implementar una estrategia de este tipo y que no fue considerado en la estimación, por lo que los beneficios potenciales de esta estrategia pueden ser aún mayores.

Es importante destacar que nuevas investigaciones deben ser orientadas hacia el análisis de posibles casos donde como resultado de la optimización se obtengan valores que no representen óptimos globales. Este problema es eventualmente encontrado en los métodos clásicos por lo que es probable encontrarlo también en métodos de búsqueda directa como el "MADS". En este caso la investigación estaría orientada a robustecer el algoritmo para el caso específico del modelado de mezclas de propiedades de hidrocarburos.

Referencias

Alex, J. Smola, & Bernhard, Schölkopf. 2004. A tutorial on support vector regression. *Statistic and computing*, **14**(3), 199–222.

Anonimo. 1997. Advanced control and information systems '97. *Hydrocarbon Processing*, **9**(76), 98–104.

Audet, C., & J. E. Dennis, Jr. 2006. Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on Optimization*, **17**(1), 188–217.

Barsamian, J. A. 2003. Contemporary problems in blending automation. *The Carmagen Engineering report.*

Bernhard, Schölkopf, Sung, K, Burges, C, Girosi, F, Niyogi, P, Poggio, T, & Vapnik, Vladimir Naumovich. 1997. Comparing Support Vector Machines with Gaussian Kernels to Radial Basis Function Classifiers. *IEEE Transactions on Signal Processing*, **45**(11), 2758–2765.

Cristianini, N., & Shawe-Taylor, J. 2000. *An introduction to support Vector Machines: and other kernel-based learning methods.* New York, USA: Cambridge University Press.

Dávila, J. A. 2011. *Lógica práctica y aprendizaje computacional.* Editorial Académica Española.

Diaz, A, & Barsamian, J. A. 1996. Meet changing fuel requirements with online blend optimization. *Hydrocarbon Processing*, Febrero, 71–76.

Eddelbuettel, Dirk, & François, Romain. 2010. RInside: Easier embedding of R in C++ applications. *Disponible en: http://dirk.eddelbuettel.com/code/rinside.html.*

Eddelbuettel, Dirk, & François, Romain. 2011. Rcpp: Seamless R and C++ Integration. *Journal of Statistical Software*, **40**(8).

Fletcher, Tristan. 2009. Support Vector Machines Explained. *Disponible en: http://www.tristanfletcher.co.uk/SVM %20Explained.pdf*, Marzo.

Forbes, J. Fraser, & Marlin, Thomas E. 1994. Accuraccy for Economic Optimizing Controllers: The Bias Update Case. *Ind. Eng. Chem. Res*, 1919–1929.

Forbes, J. Fraser, Zhang, Yale, & Monder, Dayadeep S. 2002. Real time optimization under parametric uncertainty: a probability constrained approach. *Journal of Process Control*, 373–389.

Gao, Ying, Shang, Zhigang, & Kokossis, Antonis. 2009. Agent-based intelligent system development for decision support in chemical process industry. *Expert systems with applications*, **36**(8), 11099–11107.

Gaoa, Xiaodan, Chena, Bingzhen, & Hea, Xiaorong. 2006. An agent-oriented architecture for modelling and optimization of naphta pyrolysis process. *Computer Aided Chemical Engineering*, **21**, 475–481.

Gary, James H.. & Handwerk, Glen. 2001. *Petroleum Refining Technology and Economics*. Fourth edn. CRC.

Ghalanos, Alexios, & Theussl, Stefan. 2012. *Rsolnp: General Non-linear Optimization Using Augmented Lagrange Multiplier Method*. R package version 1.12.

Gunn, Steve R. 1998 (Mayo). *Support Vector Machines for Classification and Regression*. Tech. rept. University of Southampton. Faculty of Engineering, Science and Mathematics. School of Electronics and Computer Science.

Karatzoglou, Alexandros, Smola, Alex, & Hornik, Kurt. 2011. Kernel-based Machine Learning Lab. Versión 0.9-4. *Disponible en: http://cran.r-project.org/web/packages/kernlab/kernlab.pdf*, Noviembre.

Le Digabel, Sebastien. 2011. Algorithm 909: NOMAD: Nonlinear Optimization with the MADS algorithm. *ACM Transactions on Mathematical Software*, **37**(4), 44:1–44:15.

Michalek, T.F., Nordeen, K., & Rys, R. 1994. Using a relational database for blend optimization. *Hydrocarbon Processing*, Septiembre, 47–49.

Monder, Dayadeep S. 2001. *Real time optimization of gasoline blending with uncertain parameters*. M.Phil. thesis, University of Alberta. Department of chemical and material engineering, Canadá.

Ramírez, Miguel Angel, Dávila, Jacinto, & Colina, Eliezer. 2009. Intelligent supervision of petroleum processes based on Multi-Agent Systems. *WSEAS Transaction on Systems and Control*, **4**(9), 435–444.

Ríos, Luis Miguel, & Sahidinis, Nikolaos V. 2009. Derivative-Free Optimization: A Review of Algorithms and Comparison of Software Implementations. *In: 09AICHE. Annual Meeting.*

Rizzo, Maria L. 2008. *Statistical computing with R.* Computer Science and Data Analysis Series. Chapman & Hall/CRC.

Rubio, José. 2004 (Febrero). *Modelación y optimización de mezclado de petroleo crudo con redes neuronales.* M.Phil. thesis, Centro de investigación y de estudios avanzados del Instituto Politécnico Nacional. Departamento de Control Automático, México.

Singh, A., Forbes, J. Fraser, Vermeer, P. J., & Woo, S.S. 1999. Model-based real-time optimization of automotive gasoline blending operations. *Journal of Process Control*, **10**(Marzo), 43–58.

Singh, Aseema. 1997. *Modeling and Model Updating in the Real-Time Optimizacion of Gasoline Blending.* M.Phil. thesis, University of Toronto.

Sullivan, T. L. 1990. Refinery-wide blending control and optimization. *Hydrocarbon Processing*, Mayo, 93–96.

Team, R Development Core. 2008. R: A language and environment for statistical computing. *Disponible en: http://www.R-project.org.*

Vapnik, Vladimir Naumovich. 2000. *The Nature of Statistical Learning Theory.* Second edn. Statistic for Engineering and Information Science. Springer.

Vermeer, Peter J., Pedersen, Clifford C., Canney, William M., & Ayala, John S. 1997. Blend-control system all but eliminates reblends for Canadian refiner. *Oil & Gas Journal*, Julio.

Wang, Wei, Zhang, Qiang, Li, Yankai, & Li, Zefei. 2007. On-line optimization model design of gasoline blending system under parametric uncertainty. *Mediterranean Conference on Control and Automation*, Julio.

Wen, Yu, & Morales, América. 2004. Gasoline Blending System Modeling via Static and Dynamic Neural Networks. *International Journal of Modelling and Simulation*, **24**(3).

Apéndice A

Regresión multilineal

Este apéndice describe el procedimiento implementado en el área operacional utilizada como caso estudio, para el modelado de las propiedades RON, MON y destilación ASTM D-86.

A.1. Descripción general del método

El método se basa inicialmente en la construcción de un modelo lineal para la estimación de cada una de las propiedades de una mezcla, con la receta X utilizada en su preparación. Esta estimación lineal se basa en la suposición de que la propiedad de la mezcla es obtenida a partir de la contribución de las propiedades de cada uno de los básicos de acuerdo a su proporción volumétrica descrita en la receta.

Una vez obtenido el modelo lineal, se calcula un parámetro de ajuste (ϵ) que es una función del error de estimación de cada propiedad utilizando el modelo lineal. La salida global del modelo es la suma de la estimación lineal y el parámetro ϵ.

A.2. Planteamiento formal

Se tiene un conjunto de datos reales de propiedades de mezcla P_{jl}, obtenidas a partir de recetas X_l, donde $j = 1, 2, ..., i$ y $l = 1, 2, .., w$ son los sub-índices que identifican cada propiedad y cada receta, respectivamente. También se conocen las propiedades $\widehat{P}_{jq}$ correspondientes a los básicos, donde $q = 1, 2, .., k$, es el subíndice que identifica cada básico.

Inicialmente se construyen i modelos lineales representados por funciones de la forma:

$$h_1(X) = X^T \cdot \widehat{P}_j = P^*_{jm} \tag{A.1}$$

donde P^*_{jm} es la estimación lineal de la propiedad j para una la mezcla m con receta X. Cabe destacar que el operador $(\cdot)$ representa el producto escalar.

El error de predicción e_l se calcula de la siguiente manera:

$$e_l = P_{jl} - P^*_{jl} \tag{A.2}$$

donde l representa una mezcla cualquiera de las w disponibles. El vector que contiene todos los errores de predicción es e.

El segundo modelo lineal tiene la forma:

$$h_2(X, e) = \epsilon^* \tag{A.3}$$

donde h_2 es una función que permite realizar encontrar el vector de parámetros de ajuste ϵ^*.

Finalmente la estimación final de cada propiedad a partir de una receta se consigue mediante la siguiente ecuación:

$$h(X) = h_1(X) + h_2(X, e). \tag{A.4}$$

Apéndice B

Resultados adicionales de la validación cruzada

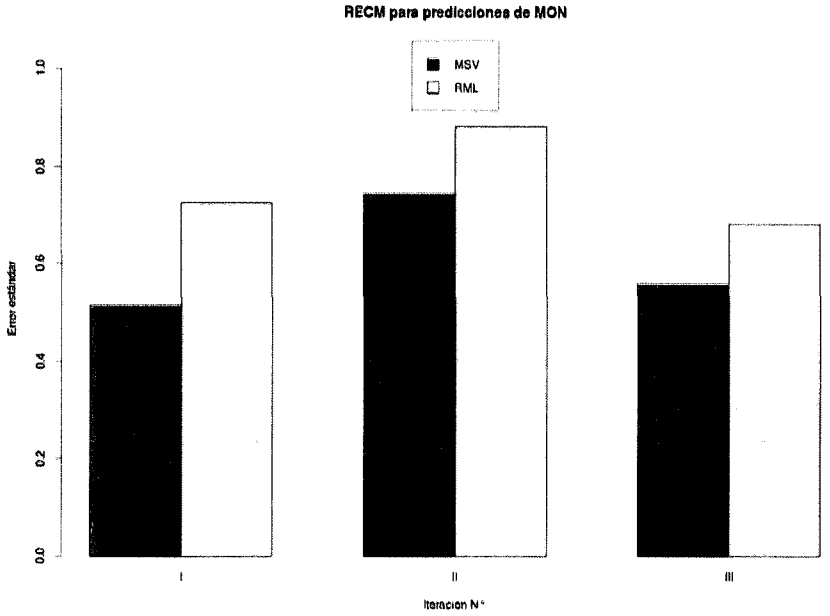

Figura B.1: RECM para MON en la validación cruzada

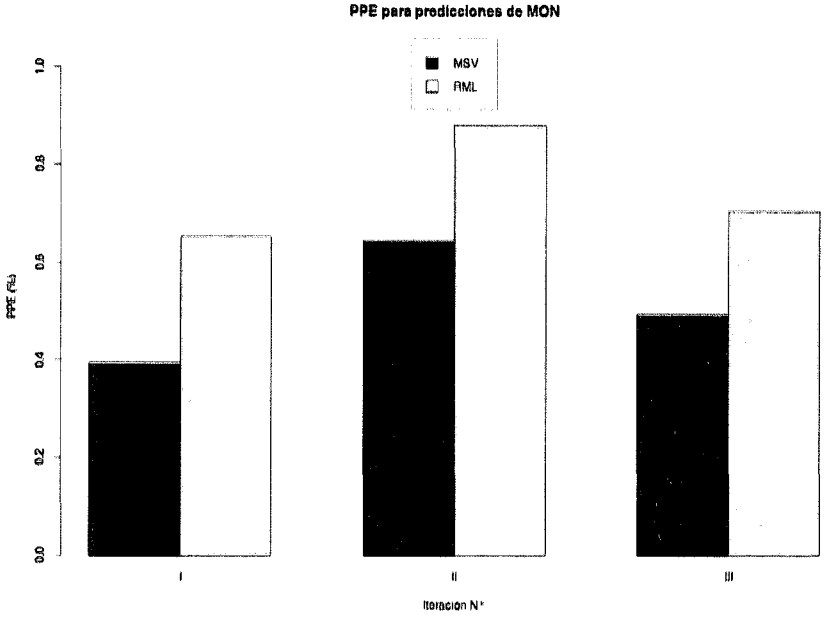

Figura B.2: PPE para MON en la validación cruzada

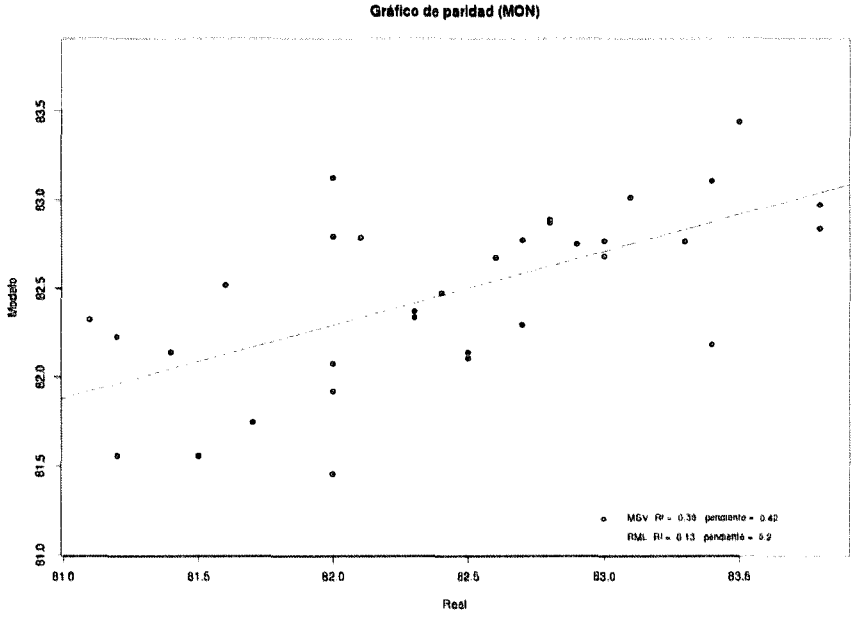

Figura B.3: Diagrama de paridad para predicciones de MON

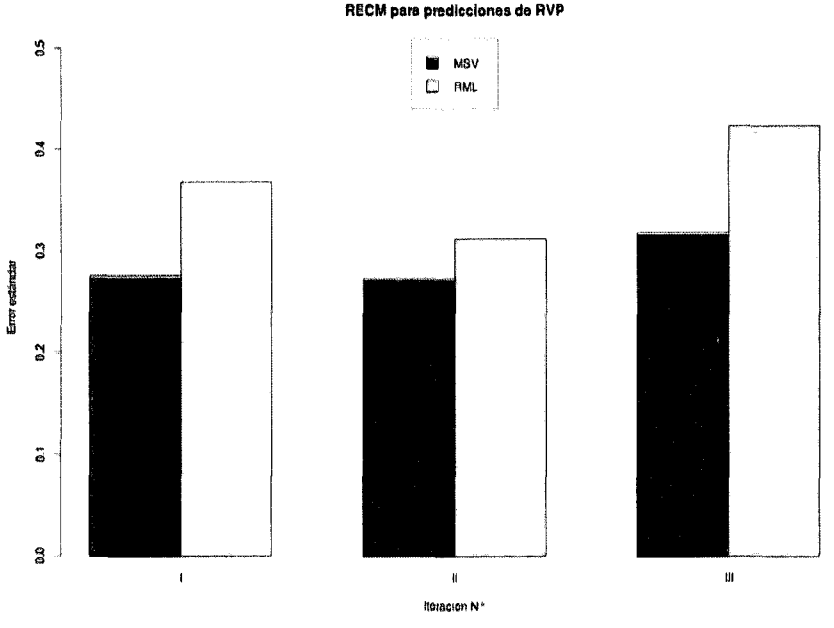

Figura B.4: RECM para RVP en la validación cruzada

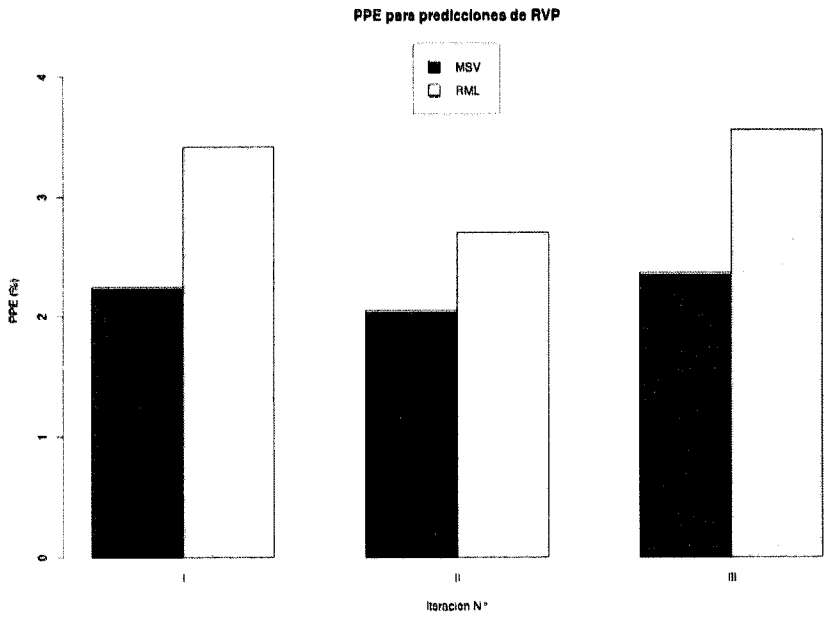

Figura B.5: PPE para RVP en la validación cruzada

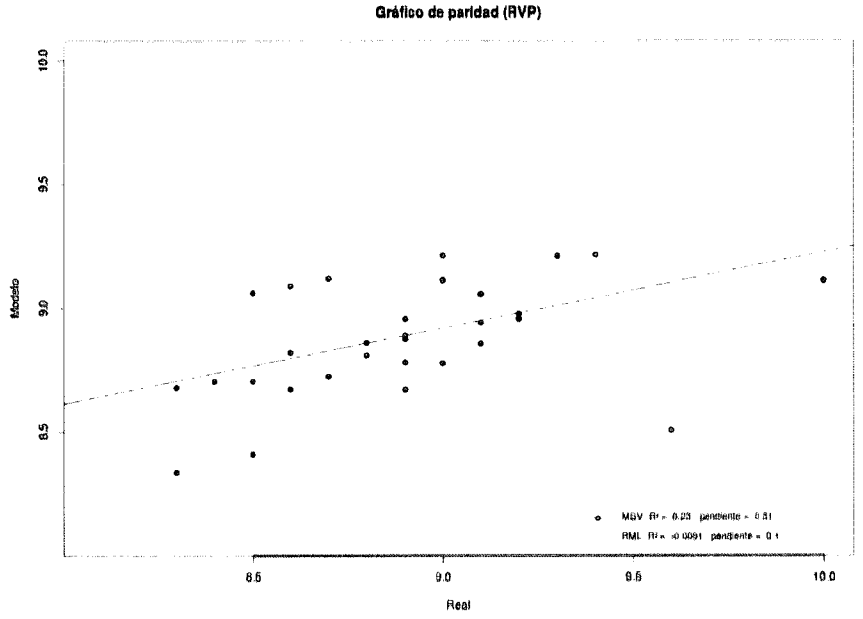

Figura B.6: Diagrama de paridad para predicciones de RVP

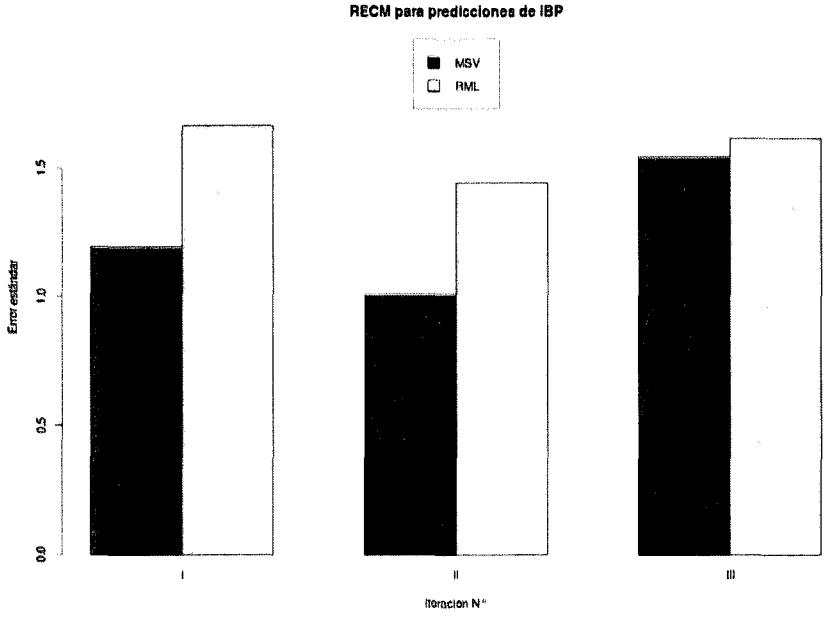

Figura B.7: RECM para IBP en la validación cruzada

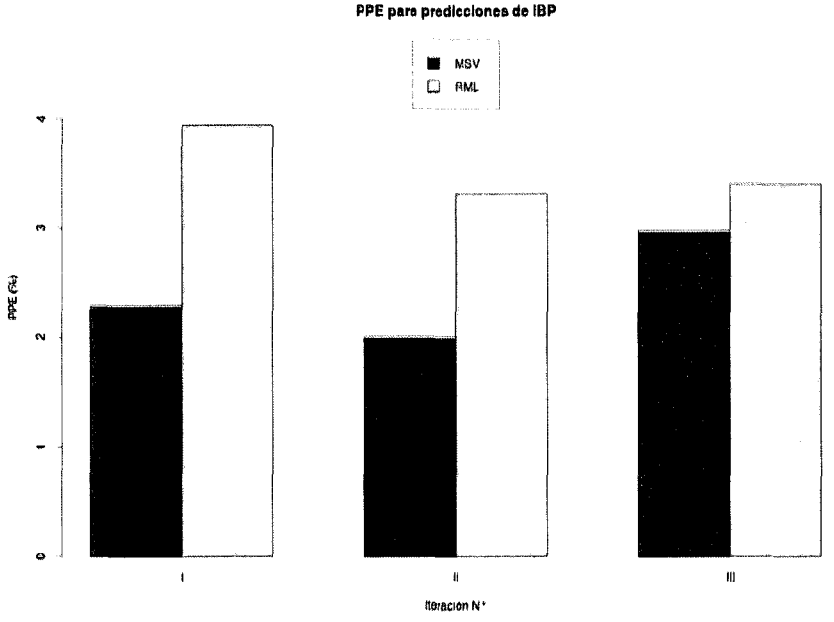

Figura B.8: PPE para MON en la validación cruzada

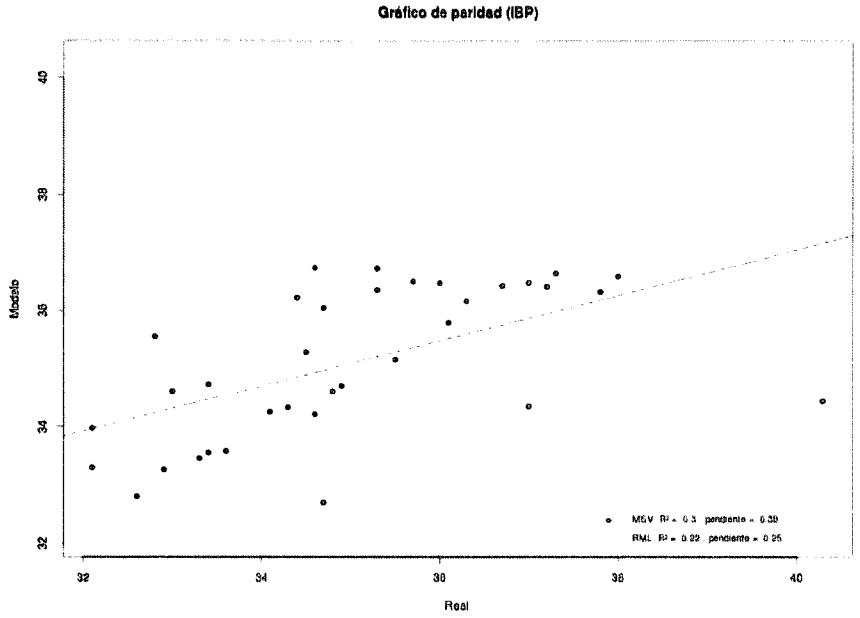

Figura B.9: Diagrama de paridad para predicciones de IBP

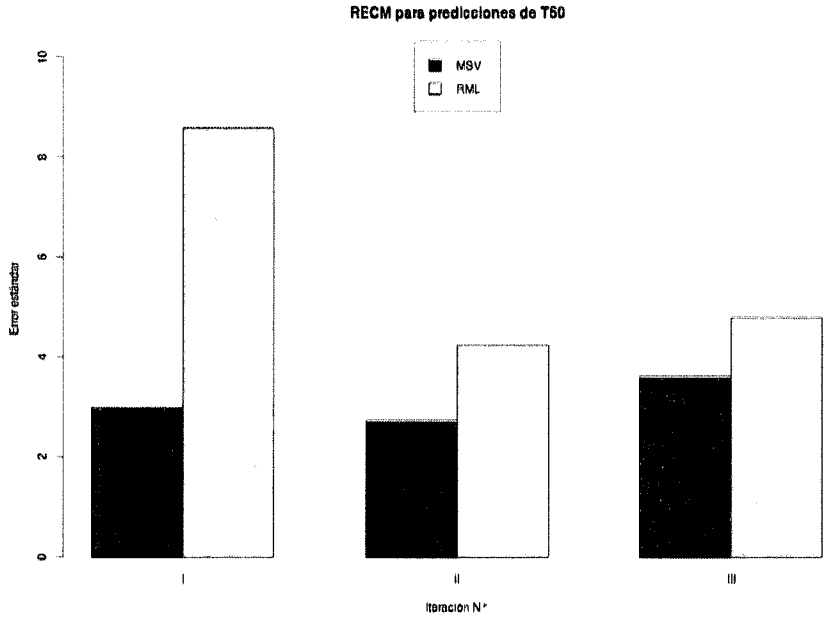

Figura B.10: RECM para T50 en la validación cruzada

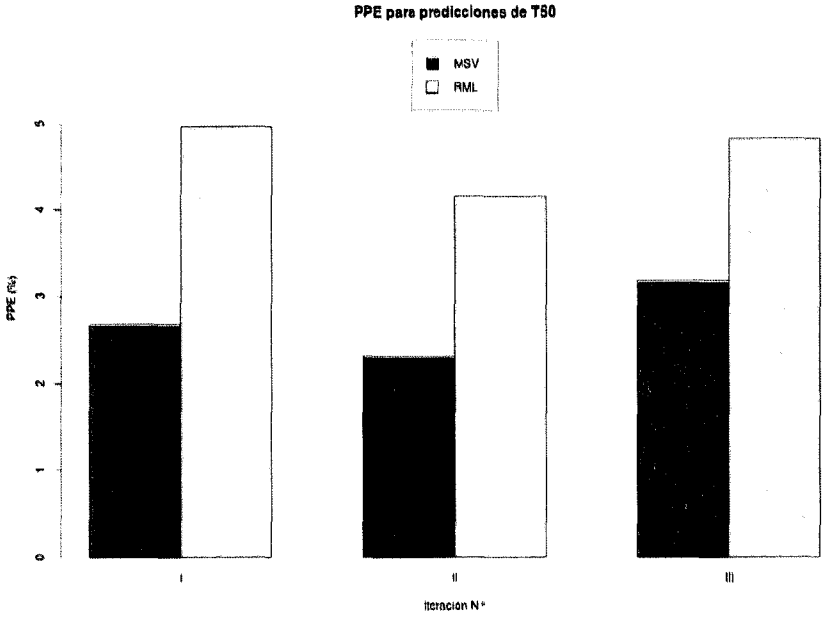

Figura B.11: PPE para T50 en la validación cruzada

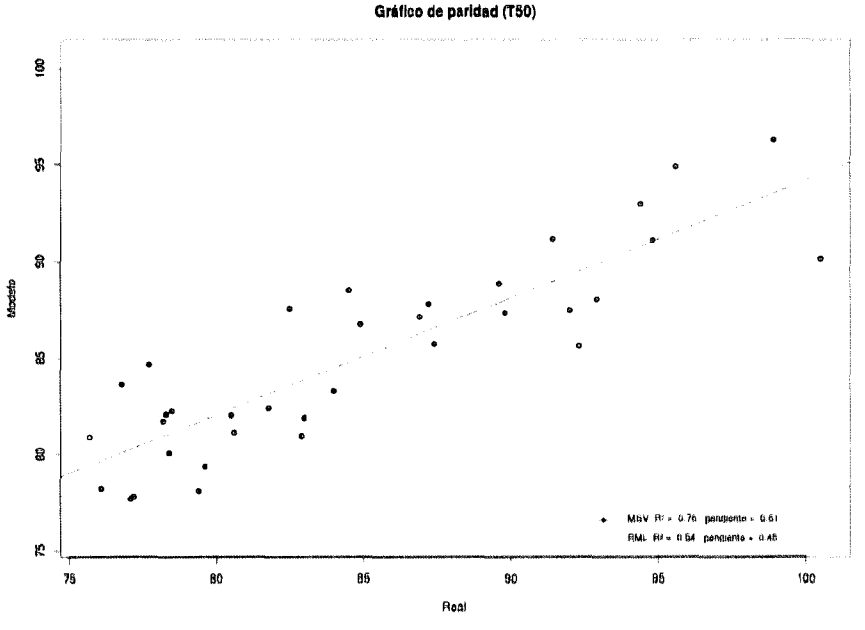

Figura B.12: Diagrama de paridad para predicciones de T50

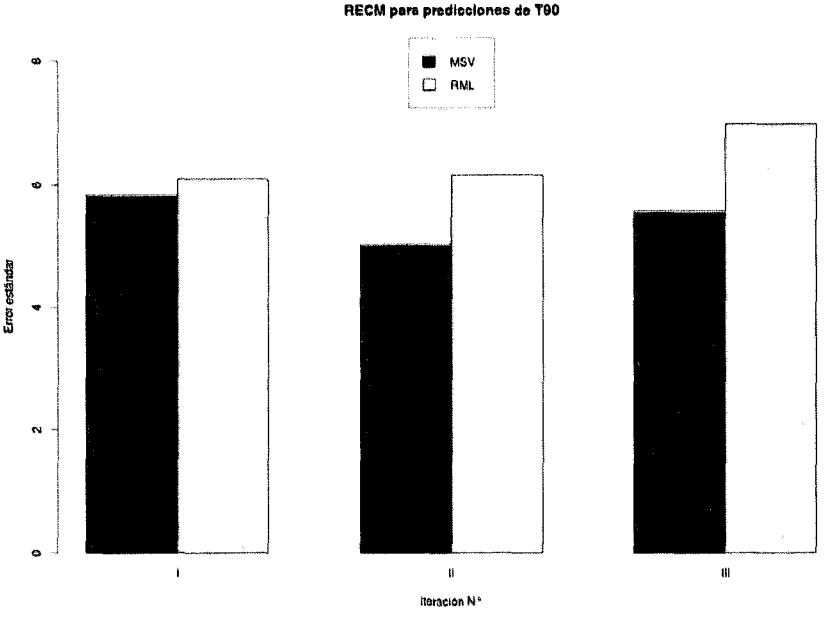

Figura B.13: RECM para T90 en la validación cruzada

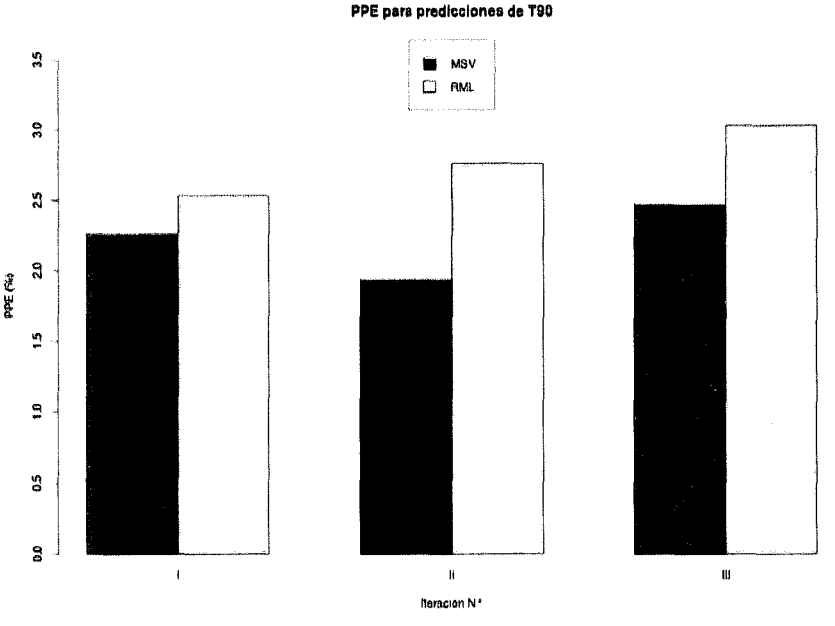

Figura B.14: PPE para T90 en la validación cruzada

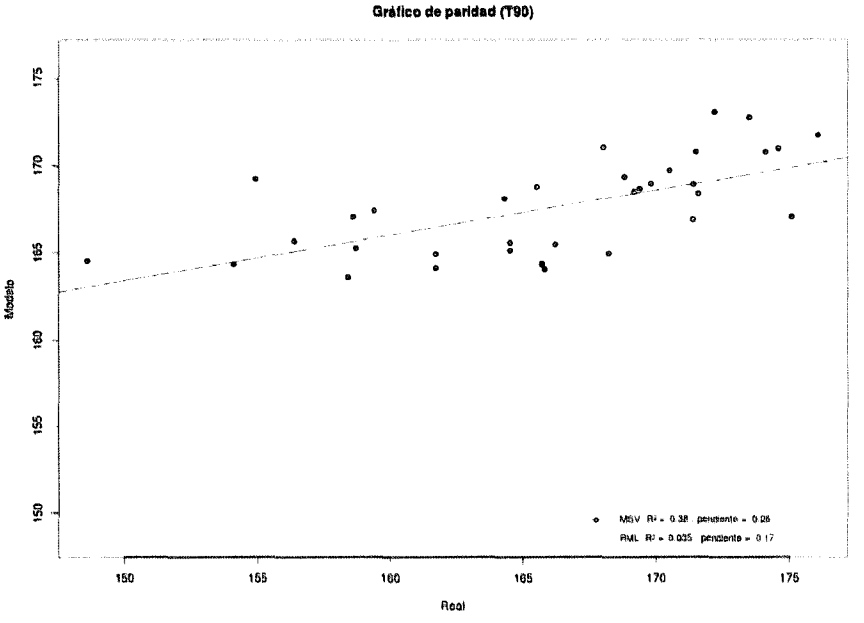

Figura B.15: Diagrama de paridad para predicciones de T90

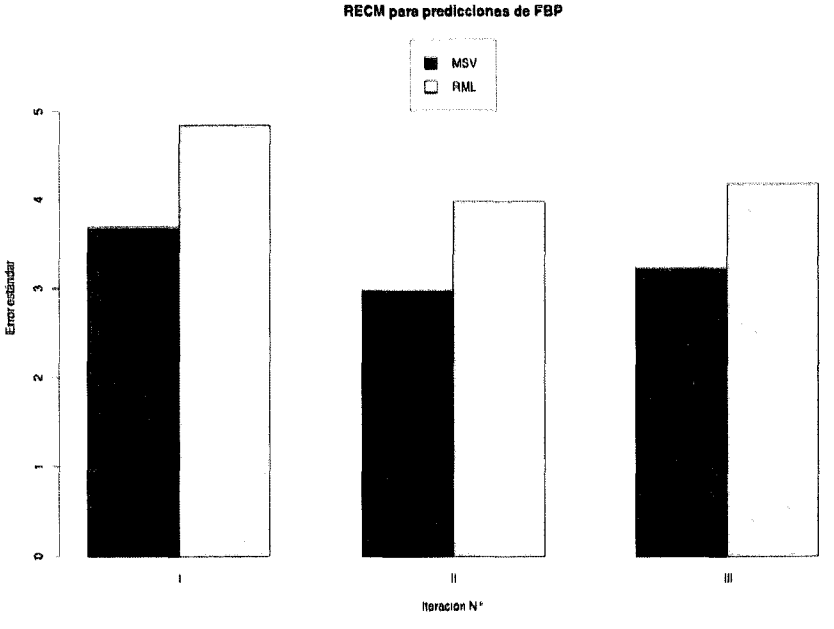

Figura B.16: RECM para FBP en la validación cruzada

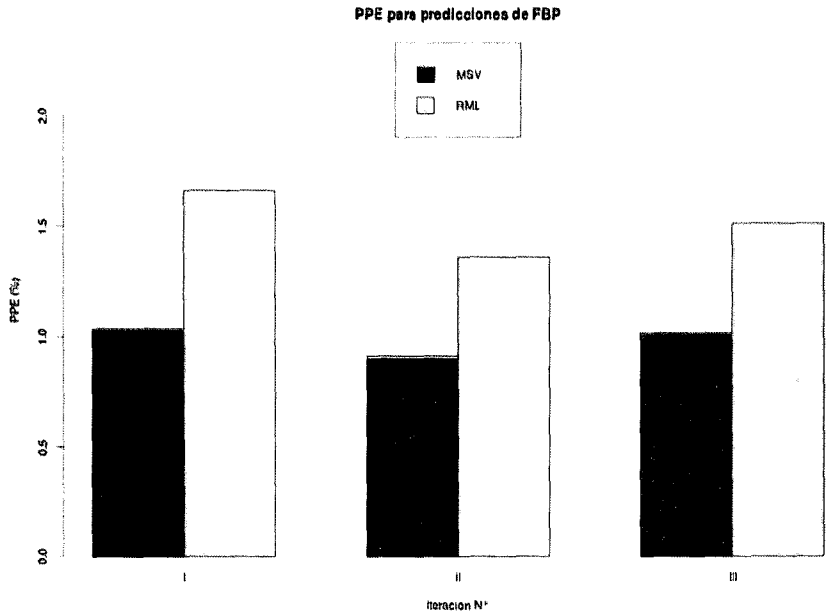

Figura B.17: PPE para FBP en la validación cruzada

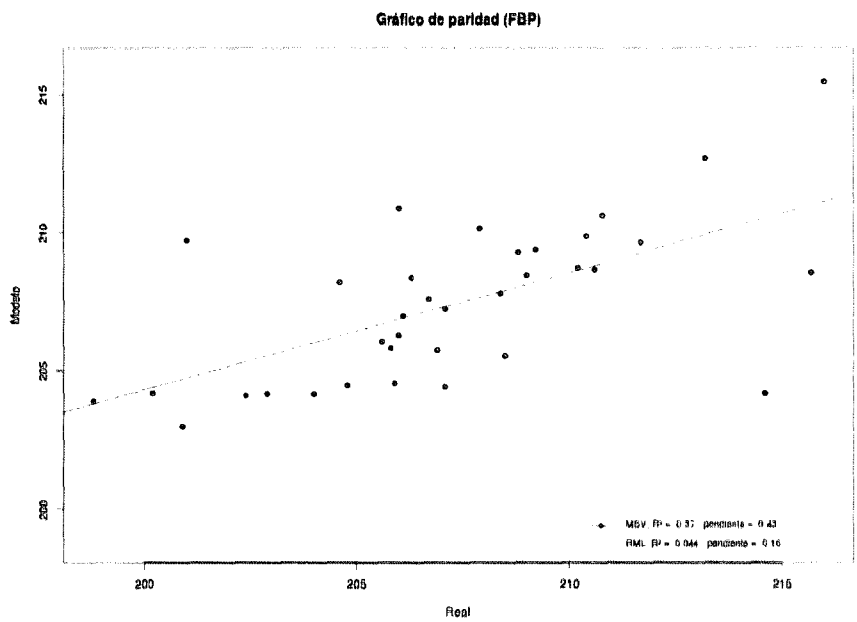

Figura B.18: Diagrama de paridad para predicciones de FBP

Apéndice C

Diagrama de objetos