Manuales de SimSEE

Volumen 6 - OddFace

Optimizador Distribuido De Funciones de Alto Costo de Evaluación.

Ing. Ruben Chaer
Abril 2019 - Montevideo - Uruguay

Sumario

1. Introducción.

En este documento se presenta la aplicación informática para Optimización Distribuida De Funciones de Alto Costo de Evaluación (OddFace). OddFace fue desarrollado en el marco del proyecto ANII FSE 18-2009 "Mejoras de la plataforma SimSEE" que finalizó en septiembre de 2012.

En pocas palabras, OddFace es una herramienta para resolver Problemas de Optimización, consistente en la búsqueda del juego de parámetros que minimiza el valor de una Función Objetivo o Función de Costo. La búsqueda se limita a los valores de los parámetros dentro de un conjunto posible denominado en la jerga matemática como Región Factible o Dominio del Problema.

OddFace se especializa en la búsqueda del mínimo en situaciones donde la Función de Costo es de alto costo de evaluación, entendiendo por tal una función que requiere un tiempo de cálculo considerable para evaluar su valor en cada punto del espacio de búsqueda (valor dado del juego de parámetros). Para ello, OddFace implementa una capa de comunicación entre posibles Robots Exploradores de forma tal de permitir la distribución del cálculo entre varias máquinas (o nodos de cálculo) manteniendo los resultados en una Base de Datos del Problema (BDP) a la que pueden acceder todos los Exploradores como se muestra en el esquema de la Fig.1. Cada Robot Explorador, consulta la BDP para conocer qué han logrado explorar los otros y en base a esa información se propone evaluar un nuevo punto (valor del juego de parámetros) del espacio de búsqueda. Dado que para la estrategia de exploración principal se implementó un algoritmo de Programación Genética, a un valor del conjunto de parámetros (o punto del espacio de búsqueda) se le llama *Individuo* y se clasifican los Individuos en la BDP de más a menos *exitosos* por orden creciente de la estimación resultante de la función de costo. Los individuos más exitosos son los de menor valor estimado de la función de costo. Los Robots trabajan en un bucle infinito hasta que se les de la orden de finalizar sus tareas ejecutando repetidas veces los pasos que se muestran en la Fig. 2

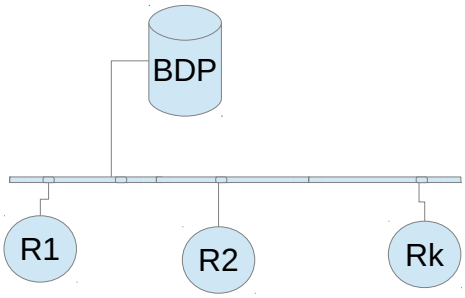

Fig. 1 Servidor de Base de Datos y Robots.

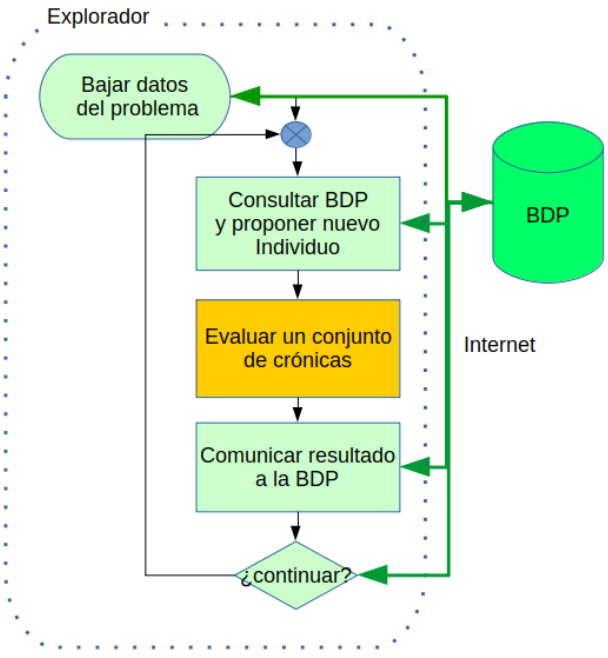

Fig. 2: Bucle de un explorador.

En el área de aplicación para la que fue diseñado OddFace, la evaluación de la Función de Costo conlleva generalmente la simulación de un sistema dinámico (un sistema donde las acciones del presente impactan sobre el futuro) operado con ciertas reglas de operación y sometido a incertidumbre durante un horizonte de tiempo. Para fijar ideas, si el problema es la optimización de inversiones en generación eléctrica, la evaluación de la Función de Costo significará simulaciones de a pasos de tiempo (por ej.

semanales) en un horizonte de decenas de años (por ej. 30 años) considerando diferentes realizaciones de los procesos estocásticos involucrados (por ej. los caudales afluentes a las centrales hidroeléctricas, la producción de energía eólica, solar, la temperatura, el precio del barril de petróleo, etc.).

En general, las incertidumbres son relevantes y para obtener resultados significativos, las simulaciones deben ser realizadas sobre un conjunto de crónicas del orden de 100 para estimaciones de valor esperado y del orden de 1000 o más cuando se trata de medir la probabilidad de eventos concretos. La Función de Costo es el valor esperado del costo futuro de operar el sistema en forma eficiente en el ensamble (conjunto) de crónicas simuladas. A mayor número de crónicas simuladas, mayor será la precisión con que se estará estimando el valor de la función de costo.

OddFace hace uso de esta evaluación en conjuntos de crónicas para llevar a cabo una búsqueda basada primero en un conjunto reducido que luego, sobre los Individuos más exitosos, es mejorada sometiendo los mismos a nuevos conjuntos de crónicas.

Como ya se mencionó, OddFace fue creado para su aplicación a la optimización del Plan de Inversiones en Generación, pero dado que resultó una herramienta eficiente para la optimización de funciones complejas, posteriormente se han desarrollado Robots para optimizar el Plan Anual de Mantenimientos, la Optimización de Agendas de embarques de GNL en base a desarrollos realizados en proyectos del Fondo Sectorial de Energía de ANII que dieron lugar a las aplicaciones OddFace_PIG, OddFace_PAM y OddFace_OptimA respectivamente. Otra aplicación de OddFace es la desarrollada en ADME para la calibración de parámetros de modelos de parques eólicos y para el entrenamiento de redes neuronales para generación de pronósticos de generación a partir de pronósticos climáticos.

Las aplicaciones principales OddFace_PIG, OddFace_PAM y OddFace_OptimA se basan en la variación de parámetros sobre una Sala SimSEE (una Sala es una representación de un sistema de generación eléctrica a simular) y la simulación con SimSEE del sistema (representado por la correspondiente Sala) permite calcular el Costo de Abastecer la Demanda (CAD) durante un horizonte temporal dado (según el problema). Es este CAD, que refleja el costo futuro de la operación óptima del sistema dado en un conjunto de realización de los procesos estocásticos representados (o crónicas en la jerga usada en el sector eléctrico) el que se utiliza como función de costo para evaluar los Individuos (con el significado que tengan según si el problema es PIG, PAM u OptimA).

La plataforma SimSEE permite realizar simulaciones del sistema de generación eléctrica de una región o país. En la simulación es posible representar tanto las centrales de generación en base a combustibles fósiles (fuel oil, gasoil, etc.) como centrales de generación hidroeléctricas (con y sin reservorios), centrales en base a fuentes renovables como la eólica o la solar, bancos de baterías, etc. También es posible representar en las simulaciones las interconexiones con otros sistemas eléctricos (por ejemplo Uruguay con Argentina y Brasil). Esta breve descripción se realiza para mostrar claramente que la simulación de un sistema tiene asociada la representación de una realidad llena de detalles e incertidumbres. Como ya se mencionó, estas simulaciones se realizan mediante la simulación de muchas "crónicas" (o "historias posibles" o "realización de los procesos estocásticos involucrados", todas expresiones que significan lo mismo en el contexto de este documento). Normalmente, como resultado principal de la simulación, se obtiene el valor esperado del Costo de Abastecimiento de la Demanda (CAD) que es la integral de los costos incurridos en el horizonte de tiempo considerado.

Un aspecto a destacar del tipo de problema de optimización, es que la Función de Costo es evaluada mediante simulaciones de Monte Carlo y, en consecuencia, lo que se obtiene es una estimación de su valor, estimación que mejora con la cantidad de sorteos utilizados. También es interesante que cada sorteo de Monte Carlo está asociado a una evaluación de la función que es

independiente del resto y esto tiene dos implicancias directas: a) permitir la evaluación distribuida de los diferentes sorteos y b) poder ir obteniendo estimadores intermedios que van mejorando con la cantidad de evaluaciones que se van realizando.

En términos formales, la función de costo a evaluar, es del tipo:

$F(X,r)$ donde $X \in D$ es el vector de parámetros de optimización del problema que puede tomar valores en el "Dominio del Problema" D y r identifica una realización posible del conjunto de procesos estocásticos. La función $F(X,r)$ es "el costo" de operar el sistema si los parámetros se fijan en X y la simulación se realiza con la crónica (suerte, o realización de los procesos estocásticos) r .

Estamos interesados en resolver el problema de encontrar "el mejor" juego de parámetros X , entendiendo por "el mejor" aquel que minimiza una función de costo que se puede construir a partir de las evaluaciones $F(X,r)$ en el conjunto de las r . A modo de ejemplo, (y seguramente el objetivo de mayor uso) es buscar minimizar el valor esperado de $F(X,r)$ en el ensamble (conjunto de realizaciones posibles) de las r .

En el caso del valor esperado, la función objetivo a minimizar es:

$$f(X) = \langle F(X,r) \rangle_r$$

En los ejemplos de simulación del sistema de generación de energía eléctrica, la función $F(X,r)$ es de "alto costo de evaluación" (poder y tiempo de cálculo) y por consiguiente los resultados de cada evaluación son almacenados en la BDP para intentar aprovecharlos al máximo e intentar no repetir evaluaciones.

En el ejemplo, considerando el sistema de generación de energía eléctrica del Uruguay, una simulación de 100 crónicas, del tipo realizada para propósitos de planificación de inversiones, con paso semanal de un horizonte de 30 años, lleva del orden de 10 minutos de cálculo en un PC de escritorio. Si hay que evaluar miles de posibles planes de expansión, la resolución en una única computadora de escritorio se torna imposible.

El algoritmo desarrollado hace una exploración del dominio D utilizando diferentes "agentes de exploración" (Exploradores) que se pueden comportar de forma diferente. Todos los Exploradores van alimentando los resultados a una Base de Datos del Problema (BDP) compartida. Los diferentes Exploradores se ejecutan en diferentes nodos de una red de cálculo, logrando así distribuir la exploración entre muchos Exploradores que trabajan en paralelo.

En cuanto a la evaluación de un punto $X \in D$, la implementación permite indicar un subconjunto del ensamble $s \in \{r\}$. Esto implica que el mismo punto puede ser calculado más de una vez, con diferentes subconjuntos de crónicas. Cada nueva información es utilizada así para mejorar la representación de la función objetivo. Si el mismo punto es evaluado en los subconjuntos de realizaciones $s_1, s_2, \ldots, s_k$, la estimación de la función objetivo irá mejorando en una serie que tiende al valor verdadero si cada nueva evaluación incluye la información de las anteriores, $f_1(X), f_2(X), \ldots, f_k(X) \rightarrow f(X)$

La exploración del dominio D se realiza entonces en forma distribuida, por varios Exploradores, utilizando la información de todas las evaluaciones compartida en forma permanente entre los Exploradores. Cada Explorador, utilizando un algoritmo de estimación de un nuevo punto (Individuo), genera una propuesta de nuevo punto de evaluación y realiza una evaluación con un conjunto de realizaciones. Se obtiene así una "estimación" de la función objetivo en dicho punto. Si el punto ya había sido calculado, se integra la nueva información mejorando la estimación existente.

Las principales componentes del algoritmo son:

1) Capa de comunicación para permitir el cálculo distribuido.
2) Exploradores de cálculo de diferentes tipos.
3) Evaluador de la función $F(X,r)$. Es este evaluador de funciones la parte del algoritmo que es necesario definir para cada tipo de problema a resolver.
4) Mejora incremental de las estimaciones.

2. OddFace_prepare – Manual de Usuario.

2.1. Introducción.

Para poder utilizar OddFace es necesario que el mismo esté instalado en un servidor. En esta sección se describe la aplicación OddFace_prepare que permite la definición de problemas para su resolución por OddFace en un servidor de cálculo.

A nivel del servidor, es posible definir Usuarios para permitir el acceso a la instalación de OddFace. A su vez, los usuarios se agrupan en Grupos que comparten la definición de los Problemas de Optimización.

En la instalación del servidor pueden agregarse las aplicaciones para resolver cada tipo de problema como ser:

OddFace_PIG = Planificación de Inversiones de Generación.

OddFace_PAM = Plan Anual de Mantenimiento.

OddFace_OptimA = Optimizador de Agendas de embarques de GNL.

En la configuración de los grupos en el servidor, se debe otorgar permiso a cada grupo sobre el tipo de problemas que puede resolver.

La aplicación “OddFace_Prepare” permite editar Problemas para su solución con “OddFace_PIG” y con “OddFace_PAM”, etc.

2.2. Ingreso (LOGIN).

La Fig.5 muestra el formulario de validación de usuario y de configuración de Perfiles de Usuario. El formulario cuenta en la parte superior con un selector de Perfiles. Los valores desplegados en las demás partes del formulario corresponden al perfil seleccionado. En el selector de perfiles, usted puede cambiar el nombre del perfil

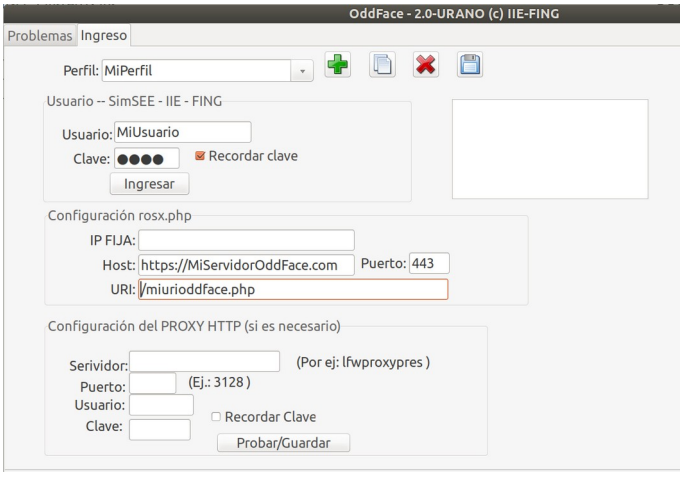

Fig. 3: Formulario de validación de usuario.

con el objetivo de guardarlo con un nombre diferente.

El Usuario y Clave deben ser los configurados para darle acceso al servidor. La información del perfil se guarda o bien presionando el botón  (reminiscencia de la época en que guardábamos en diskettes) o automáticamente al presionar el botón  .

El conjunto de botones:  permiten agregar un nuevo perfil limpio, crear un nuevo perfil a partir de copiar el actualmente seleccionado, eliminar el perfil seleccionado y guardar a disco el conjunto de perfiles.

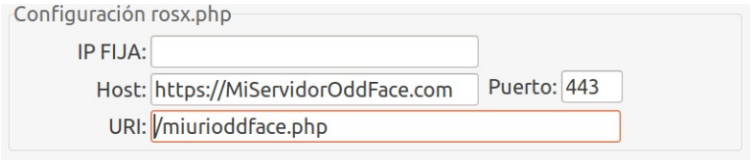

Fig. 4: Configuración de la conectividad dal servidor.

La Fig.4 muestra el panel que permite configurar la conectividad con el servidor. Como se aprecia, debe indicar el protocolo (http o https) y el host (campo Host) y el puerto correspondiente para el protocolo (Puerto, en la figura 443 puerto estándar para el protocolo https). Si se trata de un servidor con una IP FIJA y no dispone de un nombre asignado, puede utilizar el campo IP FIJA en lugar del campo Host (en cuyo caso debe quedar vacío). En el campo URI: debe poner la ubicación del script php que recibe las solicitudes OddFace. La información de este panel se la debe suministrar quién haya configurado el servidor OddFace.

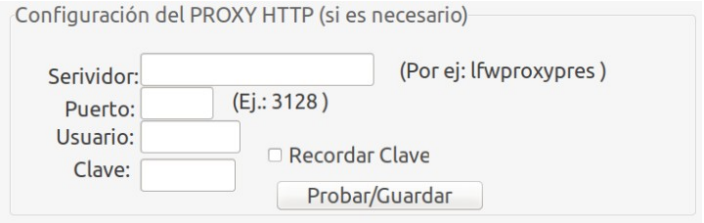

Fig. 5: Configuración de un Proxy.

Dependiendo de su ubicación, podrá necesitar configurar un Proxy para poder acceder al servidor OddFace. Para ello debe utilizar el panel mostrado en la Fig.Error: no se encontró el origen de la referencia. La información del proxy depende de la red local a la que se conecta. Como regla general, si con su navegador de internet puede acceder al servidor OddFace usando la configuración que haya colocado en el panel correspondiente a la Fig.5 debería poder configurar OddFace_prepare para conectarse también. Si no logra conectarse dejando en blanco la información del Proxy y no conoce la información del mismo, puede tratar de revisar la configuración de su navegador de internet para ver cuál es dicha configuración. El botón Probar/Guardar le permite probar si ha configurado bien el Proxy.

2.3. Listado de problemas.

Luego de Ingresar se muestra un listado de los problemas OddFace definidos. En la Fig.6 se muestra dicha pantalla. En este caso, como se puede apreciar, hay varios

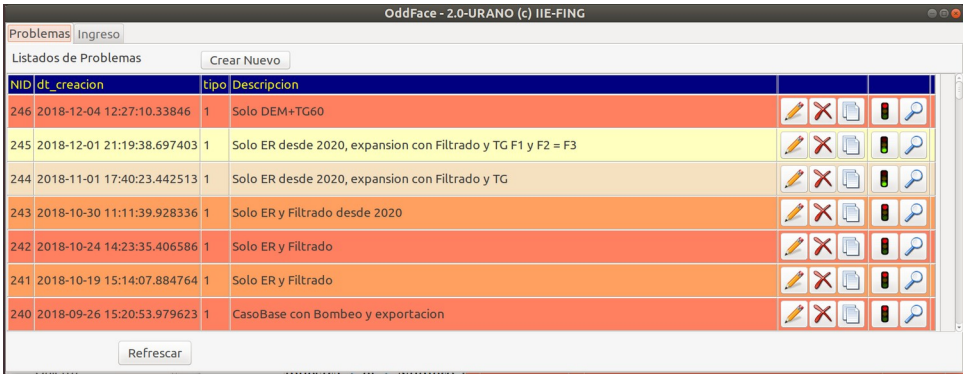

Fig. 6: Listado de Problemas.

problemas definidos. La primera columna del listado muestra el Número Identificador (NID) de cada uno de los problemas. En el ejemplo, los problemas con NID 245 y 244 tienen un color más claro que el resto. Este color más claro se debe a que dichos problemas están “activos” mientras que el resto están “inactivos”. Los botones:  le permiten “Editar los parámetros del problema (el lápiz)”, “Eliminar el problema (la cruz)”, “Crear un nuevo problema duplicando un problema (las hojas superpuestas)”, “Activar o Desactivar (según el semáforo esté en rojo o en verde respectivamente)” y “explorar los resultados del problema (la lupa)”.

Si el semáforo está en verde (problemas 244 y 245 en la Fig.6) la fila entera estará en un color más claro que si está en rojo. El semáforo en verde indica que el problema está “activo” y por tanto del lado del servidor se estará distribuyendo entre los diferentes nodos de cálculo. Si el semáforo está en rojo el problema está “inactivo” y por tanto no se estará distribuyendo para su ejecución.

Cuando se desea modificar algún parámetro en un problema existente es recomendable detener la ejecución y esperar un tiempo (alrededor de 20 minutos) para asegurarse de que todos los exploradores del Cluster se hayan detenido. De esta forma, cuando se inicie nuevamente la ejecución, los exploradores comenzarán a evaluar con los nuevos datos. Si no se detiene la ejecución del problema, los nuevos exploradores que se disparen usarán los nuevos datos, pero los que ya están en ejecución seguirán con la configuración inicial.

En la primera columna aparece el NID del problema, que es un número de identificación que se asigna automáticamente cuando se crea un problema.

En la segunda columna aparece el “dt_creación” que es la fecha y hora en la cual se creó el problema.

En la tercera columna se muestra el tipo de problema, siendo el 1 un “OddFace_PIG” y el 2 un “OddFace_PAM”.

En la cuarta columna aparece la descripción del problema introducida por el usuario durante su edición.

2.4. Edición de los parámetros de un problema OddFace.

Presionando el botón “Crear Nuevo” para crear un nuevo problema o el “Lápiz” para editar uno existente (ver Fig.6) se accede al formulario de edición de un problema OddFace que se muestra en la Fig.7.

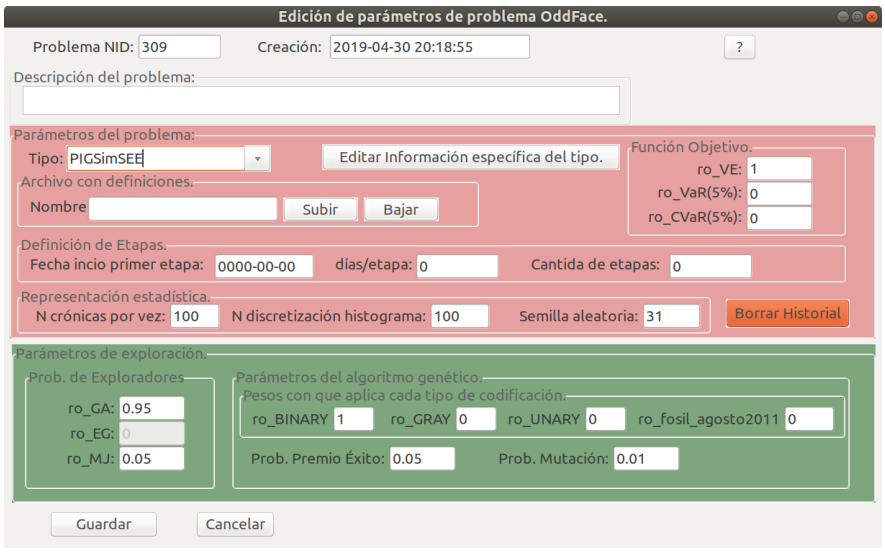

Fig. 7: Formulario de edición de un problema.

Como se puede apreciar, el formulario está dividido en tres Paneles, en el primero está la identificación

del problema, en el panel “Parámetros del problema” está la parametría principal y que define la estructura del problema y en el panel “Parámetros de exploración” están los parámetros que configuran el comportamiento de los exploradores pero que no cambian la estructura del Problema. Es importante tener en cuenta que durante la ejecución de un Problema, es posible cambiar los parámetros del panel “Parámetros de exploración” y las evaluaciones ya realizadas almacenadas en la BDP siguen siendo válidas. Por el contrario, si se cambia algún parámetro del panel “Parámetros del Problema” se estará cambiando la estructura del problema por lo cual deberá detener la ejecución y eliminar todas las evaluaciones almacenadas en la BDP (para ello utilice el botón “Borrar Historial” disponible en el mismo Panel).

El campo “Problema NID:” muestra el número identificador (NID) del Problema. El campo “Creación” muestra la fecha de creación del problema. Esto dos campos no son editables por el usaurios.

El campo “Descripción del problema” le permite al usuario introducir un texto con la descripción del problema que luego aparece en el listado de problemas (Fig.6). Es importante que introduzca un texto que le permita identificar el problema.

a) Tipo.

Este selector le permitirá seleccionar entre los tipos de problemas que el grupo al que pertenece el usuario esté autorizado a ejecutar en el servidor OddFace.

b) Editor de información del Tipo.

Este botón le permite editar información específica del Tipo de problema seleccionado. Dependiendo del tipo tendrá o no formulario de edición de parámetros adicionales. El detalle de la edición de cada tipo debe leerse en el manual correspondiente. En la Fig.8 se muestra el selector desplegado con los tipos disponibles a la fecha (mayo 2019).

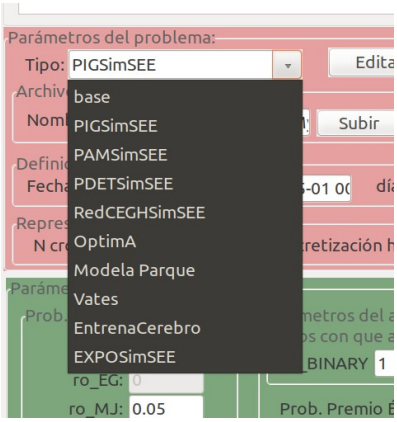

Fig. 8: Tipos disponibles (mayo 2019)

c) Archivo con definiciones.

Presionando el botón “Subir” se le permitirá seleccionar un archivo de su computadora para subir al servidor OddFace asociado al Problema que está editando. En los problemas tipo PIGSimSEE, PAMSimSEE y OptimA este archivo es una Sala SimSEE. En otros tipos de problemas, el archivo puede contener otra información. Por ej. en el problema del tipo “Modela Parque” el archivo contiene las series temporales de la estación meteorológica de un Parque Eólico y la serie temporal de la potencia generada necesaria para calibrar los parámetros del modelo.

d) Función Objetivo.

Este panel permite definir cómo se construye la función objetivo a partir del histograma de las evaluaciones $F(X,r)$ mediante asignación de pesos para combinar el Valor Esperado, Valor en Riesgo 5% (VaR = Value at Risk) y el Valor en Riesgo Condicionado a 5% (CVaR = Conditional Value at Risk). Con esos ponderadores, la Función de Costo a minimizar se forma como se muestra en la ec.

$$C=\rho_{VE}\langle F(X,r)\rangle_r+\rho_{VaR}\,VaR(F,5\%)+\rho_{CVaR}\,CVaR(F,5\%)$$ ec.(1) Función de Costo.

Donde:

- se debe cumplir que la suma de los pesos sea igual a 1 $\rho_{VE}+\rho_{VaR}+\rho_{CVaR}=1$
- $VaR(F,5\%)=v/P(F(X,r)>v)=5\%$
- $CVaR(F,5\%)=\langle F(X,r)\rangle_{5\%\,mayores}$

La Fig.9 muestra el ejemplo de la evaluación de 1000 crónicas de la operación del sistema de generación de Uruguay para los próximos 30 años actualizando los valores al 5% anual. En resumen, utilizando los parámetros ro_VE, ro_VaR y ro_CVaR puede configurar la aversión al riesgo que desea considerar. Si no quiere utilizar aversión al riesgo simplemente configure ro_VE = 1 y los otros dos parámetros a cero. (este es el uso más común).

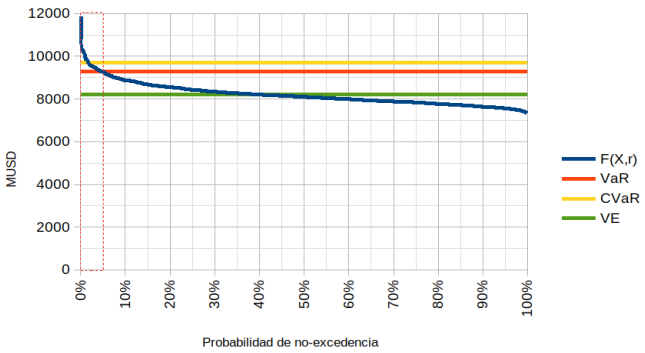

Fig. 9: Permanencia de F(X,r) y definiciones de VE, VaR y CVaR

En algunas aplicaciones OddFace, los resultados no son estocásticos (o sea la evaluación no depende de un conjunto de crónicas de simulación). En esos casos VE = VaR = CVaR con lo cual la configuración de la aversión al riesgo carece de sentido.

e) Definición de Etapas.

Si bien los algoritmos de exploración podrían aplicarse a problemas en los que no interviene el tiempo, es muy común que la optimización involucre el hecho de encontrar el mejor juego de parámetros en un conjunto de etapas temporales. Por esta razón, y orientando la solución hacia su integración con la plataforma SimSEE, se incluye entre los parámetros la descripción de un conjunto de "etapas" que deben ser consideradas como "pasos de decisión". El vector de parámetros X está compuesto por un conjunto de tramos que describen los parámetros posibles de cada etapa. Si el problema OddFace que se está resolviendo no implica etapas temporales, bastará con considerar que hay una única etapa.

- Fecha inicio primera etapa: El formato de este campo es año-mes-día (aaaa-mm-dd) usando cuatro dígitos para el año y dos dígitos para el mes y el día (formato ISO). Especifica la fecha de inicio de la primera etapa de decisión.
- Días/etapa: Este campo debe tener el número de días por etapa, es decir el número de días entre decisiones.
- Cantidad de etapas: Este campo indica la cantidad total de etapas de decisión.

f) Representación estadística.

- N crónicas por vez: Es la cantidad de crónicas utilizadas en cada simulación.
- N discretización del histograma: Es la cantidad de puntos utilizados para representar el histograma de f(x). (En esta primera versión este número debe ser igual a “N crónicas”). La idea es que pueda ser diferente, pero por simplicidad de esta primera implementación, se fijan iguales.
- Semilla Aleatoria: Se utiliza para inicializar todos los generadores de números aleatorios en los Robots específicos de evaluación de un Problema. La semilla aleatoria se impone con un valor igual al valor especificado en el formulario + cantidad de evaluaciones del individuo. De esta forma, ocurrirá siempre que la primera evaluación tiene semilla especificada en el formulario, la segunda tiene como semilla la especificada más uno y así sucesivamente. Esto permite que todos los individuos sean sometidos en su primera evaluación a la misma realización de los procesos estocásticos y por tanto permite tener una mejor estimación del desempeño del individuo (condicionado por su ADN) independientemente de “la suerte” que les toca vivir (se los somete a la misma suerte).

g) Borrar Historial.

Este botón elimina todas las evaluaciones del Problema registradas en la BDP. Es razonable que si cambia algún parámetro que cambia la estructura del problema, elimine todas las evaluaciones previas realizadas para comenzar nuevamente. Antes de Borrar Historial, también es razonable que haya desactivado la ejecución del problema (poniendo semáforo rojo al problema en el Listado de Problemas Fig.6) y que haya transcurrido el tiempo suficiente como para que todos los Robots que estuvieran trabajando sobre el problema hayan intentado comunicarse con la BDP y sean notificados que deben parar.

h) Panel “Parámetros de exploración”.

El panel “Parámetros de exploración” (color verde) contiene los parámetros que determinan el comportamiento de los Exploradores que intentan resolver el problema. Los cambios que realice en estos parámetros tendrán efecto sobre los nuevos exploradores. Si desea que todos los exploradores cambien su comportamiento lo más rápido posible, debe desactivar el problema (pasar a semáforo rojo), esperar el tiempo que demoran en ejecutarse las simulaciones más algunos minutos y luego activar el problema nuevamente. Como se mencionó, los parámetros de este panel cambian el comportamiento de los exploradores pero no cambian la estructura del problema. Por esta razón, no es necesario en este caso borrar el historial de evaluaciones.

- ro_GA, ro_EG y ro_MJ: Son las probabilidades de que actúe el mecanismo de propuesta de nuevo punto de exploración por el algoritmo genético (GA), por estimación del gradiente (EG) o que se repita la evaluación sobre uno de los puntos mejores ya evaluados con el fin de mejorar su estimación (MJ). En esta primera implementación están disponibles solamente las posibilidades GA y MJ. La implementación de EG está sin terminar, por lo cual dicho parámetro debe permanecer en 0 (cero). En el ejemplo de la Fig. 7, con probabilidad 95% se sugiere el siguiente punto a explorar usando el algoritmo genético y con probabilidad 5% se mejorará la evaluación de uno de los puntos seleccionados "como mejores". Para el criterio de selección de "los mejores" se utiliza el mecanismo de selección del algoritmo genético regulado por el parámetro GA_premio_exito que se describe a continuación.
- Tipo de Codificación: OddFace puede aplicar el algoritmo genético con diferentes formas de codificación de los parámetros. En estos cuadros se puede especificar qué tipo de

codificación se quiere usar, y para esto se debe ingresar un valor entre cero y uno en cada casilla de manera de elegir o combinar las opciones. La suma de todos los valores debe ser 1. Actualmente se encuentran disponibles BINARY, GRAY, UNARY y fosil_agosto 2011.

- ✗ *BINARY*: cada parámetro entero de optimización X / (X>=Xmin). (X <= Xmax) como la codificación binaria del número (X-Xmin). Si el parámetro es real, se especifica como Y / (Y >= Ymin). (Y<=Ymax) y la cantidad de bits a utilizar para la codificación del valor (Y-Ymin). En el caso del tipo de problema PAM, todos los parámetros son enteros. La operación de cruzamiento del algoritmo genético implica recorrer las cadenas binarias y tomar con probabilidad 50% el bit del progenitor A y con probabilidad 50% el bit del progenitor B. Sobre la cadena así obtenida, se aplica posteriormente la operación de mutación, que implica alterar algunos bits con una probabilidad dada. Dos individuos que son "cercanos" desde el punto de vista del valor de los parámetros, pueden resultar lejanos en términos de las diferencias en las cadenas binarias que representan el ADN. Por ejemplo, la cadena binaria 0001 y 1110 pueden estar representando el valor 16 y 15 respectivamente para un parámetro que puede tomar valores entre 0 y 31. El cruzamiento de 0001 y 1110 puede dar lugar por ejemplo a 0000 o a 1111 (cero y 31), con lo cual se puede apreciar que cruzar dos individuos cercanos desde el punto de vista de los parámetros lleva a individuos lejanos.
- ✗ *GRAY*: dos números consecutivos tienen diferencia de solo un bit en la codificación. Este código se utiliza para intentar reflejar la cercanía en las cadenas binarias, se utiliza el código GRAY. Cuando se utiliza la codificación GRAY, antes de realizar el cruzamiento y las mutaciones, el ADN se transforma de su representación BINARIA a código GRAY. Se realiza entonces el cruzamiento y la mutación (generando así individuos que guardan relación con sus progenitores) y luego se transforma el resultado de código GRAY a BINARIO simple.
- ✗ *UNARY*: sería el equivalente a suponer que el valor del parámetro está representado por la cantidad de 1s (unos) en un conjunto de bits. Por ejemplo, para representar números de 0 a 31 se podría suponer que se utilizan conjuntos de 310 bits (es sólo un ejemplo). Así, la cantidad de 1s (unos) en los 310 bits, dividida 10 es el valor del parámetro. Con este tipo de codificación, al realizar la operación de cruzamiento se obtiene en valor esperado un promedio entre los dos conjuntos de 1s (unos) con dispersión alrededor de dicho valor. La implementación realizada en OddFace no es exactamente una representación por conjunto de Ceros y Unos, sino que se optó por operar directamente sobre los parámetros. Dados dos individuos con parámetros X1 y X2, se genera por cruzamiento, el individuo $X=\frac{(X_1+X_2)}{2}+z(X_2-X_1)$ con $z\in U(-1,1)$ y limitando X al rango Xmin, Xmáx que corresponda.
- ✗ *ro_fosil_agosto 2011*: En esta versión de OddFace no está implementado. Se debe dejar en cero hasta que esté implementada este tipo de codificación.

• GA_premio_exito: Este parámetro regula el comportamiento de selección de individuos del algoritmo genético. En cada paso, el explorador selecciona dos individuos como los progenitores para cruzarlos y proponer un nuevo individuo. Para la selección de los progenitores parte de una lista ordenada de los individuos evaluados (registrados en la BDP), ordenados por orden decreciente de mérito (creciente del valor de “f_Objetivo”). Con probabilidad “GA_premio_exito”, se selecciona al primer individuo, y con (1-GA_premio_exito) no. Si no se selecciona el primero, se repite el mismo procedimiento con la lista, quitando el primero y así sucesivamente. Esto lleva a que se seleccionen los mejores con mayor probabilidad. La probabilidad de seleccionar el más exitoso es $GA_{prob_{exito}}$, la del

el siguiente $(GA_{prob_{exito}}-1)GA_{prob_{exito}}$ y por inducción, la probabilidad de seleccionar el $k-ésimo$ individuo en la lista ordenada por orden creciente de éxito es: $(GA_{prob_{exito}}-1)^{k-1}GA_{prob_{exito}}$.

- GA_prob_mutación: Luego de seleccionados los progenitores y cruzados por simple combinación azarosa de sus genes, se aplica el mecanismo de "mutación" que invierte bits de la cadena binaria que representa el ADN del individuo. “Prob. Mutación”, es la probabilidad de mutación de bit de la cadena de ADN. En la medida que un problema tenga una cadena de ADN más larga, la probabilidad de que ocurra alguna mutación es mayor para igual parámetro “Prob Mutación”. En la implementación, dada la dimensión N de la cadena de bits del ADN, se construye una fuente aleatoria que determina la cantidad m de secuencias de mutación a aplicar como $p_N(m)=C_m^N P_{Mut}^m (1-P_{Mut})^{(N-m)}$. Dada una cadena de ADN (secuencia de bits) se utiliza esta fuente para determinar la cantidad m de secuencias de mutación a imponer. Cada secuencia de mutación, se aplica seleccionando con distribución uniforme en la cadena de bits un bit e invirtiéndolo. Se repite este proceso m veces. Puede resultar que se invierta nuevamente un bit antes invertido.

2.5. Consultas sobre el historial de evaluaciones.

El historial de evaluaciones se almacena en una tabla en la BDP compartida por todos los nodos de cálculo.

Para consultar el historial se puede utilizar el botón  del listado de problemas (Fig.6). Al presionar ese botón se despliega el formulario de consultas que se muestra en la Fig.10. El panel “Consulta SQL” permite la edición de los parámetros de una consulta SQL a realizar sobre la tabla del historial de evaluaciones del problema. Como se puede apreciar, los campos “nid”, “adn”, “f_Objetivo”, y “cnt_evaluaciones” están seleccionados por defecto. En el cuadro de edición a continuación, usted puede agregar campos adicionales. Los campos de la tabla están listados en el cuadro de la derecha del mismo panel.

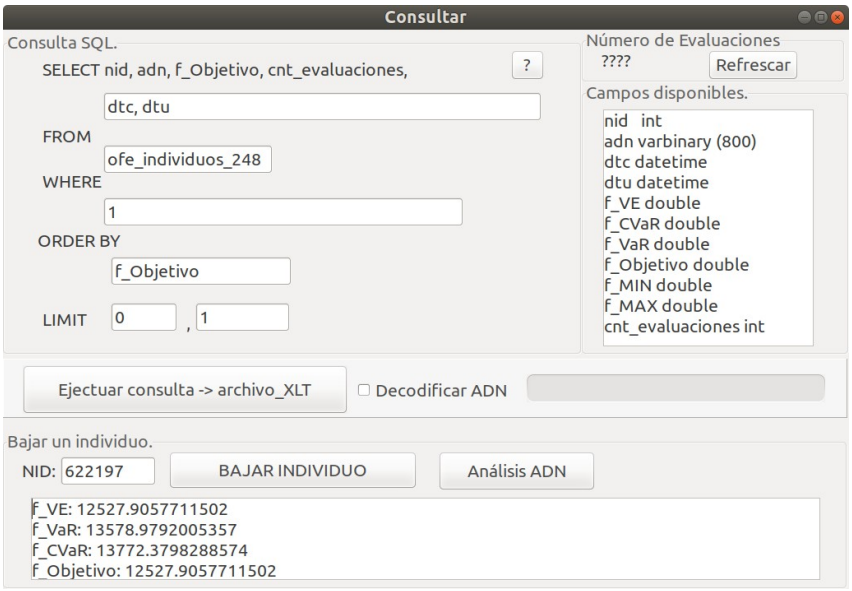

Fig. 10: Formulario de consulta de resultados.

Los significados de los campos son los siguientes:

- "nid", número identificador único del punto evaluado.
- “adn”, es el adn del individuo (número binario resultante de los cruzamientos y mutaciones) y se muestra en código hexadecimal.
- "f_Objetivo", valor de la función objetivo en el punto evaluado.

- "dtc", fecha y hora de la primera evaluación del punto.
- "dtu", fecha y hora de la última evaluación del punto.
- "cnt_evaluaciones", cantidad de evaluaciones realizadas del punto.
- “f_VE”, “f_VaR”, “f_CVaR” corresponden al valor esperado, al Valor en Riesgo 5% y al Valor en Riesgo condicionado al 5% del conjunto de crónicas en que fue evaluado el individuo.
- “f_Objetivo” es el valor de la función de costo objetivo de la minimización.
- “f_MIN” y “f_MAX” son los valores mínimo y máximo de costo correspondiente al conjunto de crónicas evaluadas.

El casillero FROM tiene el nombre de la tabla del historial del problema y no puede ser editado.

El casillero WHERE tiene el "filtro" y por defecto tiene 1, lo que significa que todos los registros serán seleccionados. Por ejemplo, si se pusiera "cnt_evaluaciones” > 4 en el casillero WHERE, sólo se seleccionarán aquellos puntos que cuentan al menos con 5 evaluaciones.

El casillero ORDER BY indica el orden en el que se ordenarán los resultados. En el ejemplo, están ordenados por orden decreciente del número identificador único (nid DESC), lo que significa que aparecerán por orden inverso de creación, o sea, al inicio los últimos creados.

Los casilleros a la derecha de LIMIT fijan el desplazamiento desde el primer registro y la cantidad de registros a bajar. En el ejemplo de la figura, los valores son 0 (cero) desplazamiento y 10000 (diez mil) registros.

Una vez establecidos los parámetros que permiten formar la consulta SQL, presionando el botón "Ejecutar consulta -> archivo_XLT" se envía la consulta al servidor y se recibirán los registros que resulten seleccionados guardados en un archivo .xlt (archivo de texto con números con un “.” como separador de decimales y separados por tabuladores).

El panel "Bajar individuo" le permite especificar el "NID" del individuo que desea bajar. Presionando el botón “Bajar Individuo”, se baja el ADN del individuo identificado por el NID especificado y se ejecuta localmente el Robot de evaluación para decodificar el ADN y crear la representación del mismo que corresponda. Por ej., en los problemas del tipo PIGSimSEE se ejecuta localmente la aplicación “oddface_pig” para crear la sala SimSEE correspondiente al Individuo bajado.

Para ello, en la computadora en la que se está ejecutando "OddFace_prepare" debe estar instalada también una versión de la aplicación correspondiente al tipo de problema que esté resolviendo para que se pueda ejecutar sobre el ADN bajado y crear la representación que corresponda. Cuando se realiza esta consulta se deben verificar los mensajes en la pantalla de consola de Oddface, pues la aplicación que se ejecuta puede escribir información relevante o notificar de un fallo. En el caso de los problemas PIGSimSEE se escribe en la consola el lugar en que se creó la

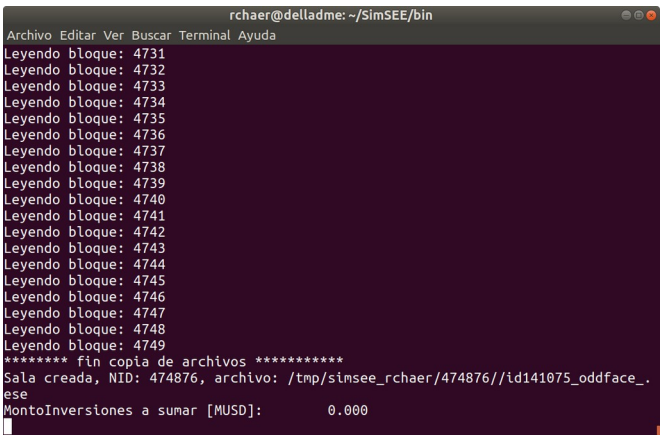

Fig. 11: Ejemplo de escritura en la salida de texto al bajar un Individuo.

Sala que representa al individuo y el monto total de inversiones actualizado que no está representado en la Sala. Para que pueda leer las salidas de la consola, luego de finalizar se queda esperando a que el usuario presione “Enter”, luego de lo cual se puede retornar al formulario de consultas.

La sala se descarga en una carpeta en el directorio temporal (su ubicación depende según que se trate Linux o Windows), con el nombre original de la Sala concatenado con el texto: “_oddface_”. La Fig.9 muestra un ejemplo de descarga del individuo en el caso de un problema PIGSimSEE.

Cada tipo de problema debe implementar dónde guarda el Individuo decodificado y cómo lo notifica en el terminal de texto.

El botón "Análisis ADN" permite ver la cadena binaria de un individuo. Para esto se debe ingresar el NID del individuo que se quiera ver y luego presionar ese botón. Puede ser necesario hacer “click” en el cuadro de texto para que aparezcan los resultados. La Fig.12 muestra un ejemplo del resultado de presionar dicho botón. Esta funcionalidad se implementó más que nada con propósitos de depuración del código y de investigación sobre la forma de codificar los problemas. En el uso corriente de OddFace difícilmente necesite utilizarla.

En primer lugar se muestra la cadena tal cual está en la base de datos, luego se muestran los valores decodificados y después el ADN vuelto a codificar, pero "limpiando los bits redundantes".

Se obtiene un representante principal del conjunto de ADNs que decodificados tienen el mismo resultado. Las tres cadenas de ADNs mostradas debieran ser iguales si en ningún momento se opera sobre los bits redundantes y por eso esta funcionalidad es solo para depuración de las implementaciones.

memo_ADN
nbits_Justo: 403, nbits_Resto: 13
000000100000000000001000000000000100000000000000000000000000000000000000000000001000000000000000010000000000000001000001000000000000
000000000000000010000000001000000000001000101000000100001000010000000000000000001100000000000100000000000100000000010100101000000000
000000001001000000100000000100000000000000010000010000000100000000001000000000000010010010000100010000000001000000000000000000000001
10111110000000000000
, 0, 2, 0, 0, 4, 0, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 1, 0, 0, 8, 0, 2, 0, 0, 0, 0, 0, 0, 1, 0, 4, 0, 0, 2, 5, 0, 1, 1, 1, 0, 0, 0, 16, 1, 0, 16, 0, 0, 8, 0, 0, 5, 5, 0, 0, 0, 4, 4, 16, 0, 0, 1, 0, 0, 2, 4, 0, 4, 0, 0, 4, 0, 0, 8, 2, 2, 2, 1, 0, 2, 0, 0, 0, 0, 0, 27, 5
000000100000000000001000000000000100000000000000000000000000000000000000000000001000000000000000010000000000000001000001000000000000
000000000000000010000000001000000000001000101000000100001000010000000000000000001100000000000100000000000100000000010100101000000000
000000001001000000100000000100000000000000010000010000000100000000001000000000000010010010000100010000000001000000000000000000000001
10111010000000000000
000000100000000000001000000000000100000000000000000000000000000000000000000000001000000000000000010000000000000001000001000000000000
000000000000000010000000001000000000001000101000000100001000010000000000000000001100000000000100000000000100000000010100101000000000
000000001001000000100000000100000000000000010000010000000100000000001000000000000010010010000100010000000001000000000000000000000001
10111010000000000000
nid: 474876
cnt_evaluaciones: 1
f_VE: 12773.573529928
f_VaR: 13824.4575875048
f_CVaR: 14003.188216767
f_Objetivo: 12773.573529928
f_Histo: [100| 11929.158272361; 11968.7777829425; 12005.0194718083; 12017.1565708214; 12032.3833775757; 12046.5615344235; 12081.1109117633; 12085.7349566305; 12117.7680887019; 12169.1434718785; 12190.7753051286; 12201.7096846919; 12231.4795699569; 12315.7410197828; 12327.1029200248; 12334.0501517377; 12351.1219800798; 12353.4087861756; 12354.9507164018; 12375.0199592596; 12377.4724869632; 12381.5574325203; 12384.2276541194; 12389.1539730761; 12392.1638360465; 12394.1115757286; 12402.3142867478; 12403.4885837379; 12422.1193583345; 12424.1361327088; 12425.1697896231; 12434.3190790531; 12453.4059284225; 12460.5341698898; 12466.7354772204; 12483.5152230585; 12495.9436812823; 12496.8126665566; 12517.6572022949; 12524.4382285267; 12550.6358545089; 12574.6108930083; 12580.7697237213; 12588.3279631149; 12598.2770233897; 12606.5198716438; 12609.5334123862; 12652.5680431197; 12653.752998765; 12659.1172450018; 12682.2999383909; 12710.1392650936; 12720.7142861705; 12734.1520370324; 12780.5459331089; 12805.2450617702; 12826.9862398139; 12830.0526367755; 12841.484460638; 12850.7652707411; 12864.5032420755; 12875.5479192703; 12885.3947376688; 12908.5903672232; 12920.4805401952; 12923.2470005674; 12923.4269463457; 12956.4132878106; 12975.0591853484; 12990.465039251; 13004.9011967463; 13007.6453164042; 13021.9359964188; 13022.7502548968; 13025.8926802639; 13035.1708786129; 13072.4518315991; 13085.142761231; 13101.9221390161; 13115.6868438361; 13137.7838024691; 13175.402268806; 13181.2429776286; 13184.310983076; 13267.5292241882; 13319.5517343409; 13338.6656450099; 13341.0955374053; 13347.631333053; 13430.4558515479; 13612.0657577478; 13663.6720406938; 13754.6268974598; 13793.6120177385; 13824.4575875048; 13913.9685338068; 13928.4262863502; 13973.9849927549; 14143.9072811609; 14234.3846190244]

Fig. 12: Análisis ADN

En la Fig.12 se muestra un caso en el cual nbits_Justo: 403, nbits_Resto:13. En este ejemplo, la cantidad de bits mínima necesaria para codificar un individuo es 403 bits y el resto de 13 bits son los sobrantes para codificar las cadenas de ADNs con secuencias de palabras de 16 bits

3. Detalles de la implementación de los Algoritmos Genéticos en OddFace.

En esta sección se describe la implementación de los Algoritmos Genéticos en OddFace al nivel suficiente como para poder entender y realizar modificaciones del código.

La implementación está repartida en dos Units (Módulos Pascal) que son “utipos_ga” en la que se definen los tipos básicos de datos para la representación de las cadenas de ADN y de los Genotipos y “uoddface” en la que se definen las clases bases para la descripción de problemas de optimización OddFace.

3.1. Definiciones básicas para manejo de cadenas de ADN.

La unidad Pascal "utipos_ga" contiene las definiciones básicas para el manejo de Algoritmos Genéticos. En esta sección se usará ***“Negrita cursiva”*** para escribir definiciones textuales del código fuente Pascal.

Lo primero a resaltar, es que en la implementación las cadenas de ADN son representadas por vectores de BITs organizados vectores de datos de 16 bits (tipo WORD de Pascal). O sea que toda cadena de ADN está almacenada como vectores de palabras de 16 bits.

En la Interface, se define el tipo:

TCadenaADN = packed array of word;

Así definida, una instancia de TCadenaADN será un vector de longitud a definir de datos del tipo *Word* (16 bits Pascal). La palabra “packed” le indica al compilador que ubique en forma “contigua” todos los bytes, previendo que podremos acceder a los mismos en forma directa. Igualmente en la implementación se buscó acceder siempre por medio de la estructura del vector, con lo cual no debiera ser relevante el que los bytes estén contiguos en memoria.

Con esta definición de ADN, si se quiere acceder al bit *k* de una cadena, hay que acceder al dato (*k div 16*) del vector y dentro de ese dato al bit (*k mod 16*). Para acceder al bit (*k mod 16*) hay que construir una máscara de bit que tenga 0 en todos los bits salvo en la posición (*k mod 16*) para lograr así, mediante una operación binaria AND, aislar el bit al que se quiere acceder.

Se definen las constantes:

- ***BIT_MAS_SIGNIFICATIVO = $8000;*** Esta constante sirve de máscara para aislar el bit más significativo en un dato de 16 bits.
- ***BIT_MENOS_SIGNIFICATIVO = $0001;*** Esta constante sirve de máscara para aislar el bit menos significativo en un dato de 16 bits.

Entonces, dado un vector de datos ADN para leer el bit "k" (suponiendo k: 0.. NBits-1) tendríamos que hacer:

mascara_de_bit = BIT_MENOS_SIGNIFICATIVO shl (k mod 16);

jw:= k div 16;

Bitk:= ADN[jw] and mascara_de_bit;

Donde: el operador "*shl*" es estándar de Pascal (se lee Shift Left) y tiene el efecto de hacer un desplazamiento binario hacia la izquierda del primer parámetro (*k mod 16*) posiciones.

El operador "*div*" es estándar de Pascal y calcula la parte entera de la división.

El operador "*mod*" es estándar de Pascal y calcula el residuo de la división.

Para poner a UNO el Bit k de la cadena ADN la operación sería:

ADN[jw]:= ADN[jw] or mascara_de_bit;

Para poner a CERO el bit k de la cadena ADN la operación sería:

ADN[jw]:= ADN[jw] and NOT(mascara_de_bit);

Para invertir el bit k de la cadena ADN la operación sería:

ADN[jw]:= ADN[jw] xor mascara_de_bit ;

a) ADN-> Genotipo y Fenotipo.

Toda la información contenida en los cromosomas se conoce como genotipo, sin embargo dicha información puede o no manifestarse en el individuo. El fenotipo se refiere a la expresión del genotipo más la influencia del medio.

La Cadena de ADN es una representación del Genotipo y caracteriza totalmente al individuo desde el punto de vista genético. La experiencia posterior de cada individuo en su ambiente podrá llevar a diferenciar individuos que desde el punto de vista genético son idénticos. Por ejemplo, dos gemelos con igual información genética, luego son diferenciables por rasgos externos.

Los términos "genotipo" y "fenotipo" fueron creados por Wilhelm Johannsen en 1911. El genotipo es la información hereditaria completa de un organismo, incluso si no se expresa. El fenotipo es una propiedad observada en el organismo, como la morfología, el desarrollo, o el comportamiento.

Observando la cadena de ADN se puede conocer el genotipo, observando la apariencia y desempeño del organismo en su ambiente se puede conocer el fenotipo.

Las propiedades físicas de un organismo son las que determinan directamente sus posibilidades de supervivencia y reproducción, mientras que la herencia de las propiedades físicas sólo se produce como una consecuencia secundaria de la herencia de genes.

El mapeo de un conjunto de genotipos a una serie de fenotipos a veces se denomina mapa genotipo-fenotipo. En el caso de aplicación en OddFace sobre la plataforma SimSEE, el genotipo de un individuo determina todo lo necesario para poder simular y evaluar la performance de ese individuo. Por ejemplo, fija todos los parámetros de una Sala SimSEE para hacer la simulación y obtener el costo futuro de operación del sistema como un índice de desempeño. Al realizar una simulación, se seleccionará un conjunto de crónicas (muestreos de Monte Carlo) y es así que dos evaluaciones del mismo individuo con dos conjuntos de crónicas diferentes puede llevar a dos evaluaciones diferentes de la performance del mismo individuo. Es así que en el caso de aplicación, que involucra una plataforma de simulación que simula “el ambiente” en el que el “individuo” (caracterizado por su genotipo) tiene la oportunidad de desempeñarse y mostrar así su Fenotipo.

Esta representación, ajustada a la realidad en que un mismo genotipo puede correr con

suertes diferentes durante la evaluación, puede considerarse como un comportamiento “robusto” de la naturaleza, en que no se define un “líder absoluto” sino que existen un conjunto de individuos que son “los mejores”, pero que dependiendo de la suerte de vida de cada uno, hay algunas genéticas que resultan mejores, pero en otra circunstancia puede invertirse la condición de ser mejor. Esto de alguna forma impide que quede una genética ÚNICA como ganadora, y es importante para que no se pierda la capacidad de adaptación de los organismos. Pero pensando en el objetivo de encontrar el individuo óptimo como la solución a un problema concreto que tiene una especificación clara (no condiciones externas que van cambiando) esta influencia de “la suerte” del individuo durante la evaluación es más bien un aspecto negativo de los AG. En esta primer implementación de OddFace v1.0 se ha implementado que “no hay control” sobre la suerte del individuo y por lo tanto, el mismo genotipo, evaluado dos veces puede tener diferenciación. En el análisis de los casos de test realizados para esta implementación más adelante se profundiza sobre este tema y se propone una implementación alternativa, que aunque no sea ajustada a la realidad, se considera mejor para la resolución del tipo de problemas de test realizados.

b) Descriptores de Genotipos.

En la Unidad “utipos_ga” se define la clase:

```
TDescriptorGenotipo = class
  nombre: string; // identificador del parámetro
  nbits: integer;
  constructor Create( nombre_: string; nbits_: integer );
  procedure codificar_ADN( var adn: TCadenaADN; var offset: integer; var mask: word; var Genotipo ); virtual;
  function decodificar_ADN( var Genotipo; var adn: TCadenaADN; var offset: integer; var mask: word ): boolean; virtual;
end;
```

Esa clase se utiliza para describir un Genotipo, o parámetro del individuo. Por ejemplo, en el caso del PAM, dada una Orden de mantenimiento de una unidad (Por Ej.: “Sacar la 6ta Unidad de Central Batlle durante 15 días para mantenimiento entre 1/1/2012 y 1/3/2013) el genotipo podría representar para esa orden la fecha de inicio del mantenimiento.

En el descriptor del Genotipo:

- La propiedad “nombre” nos permite identificar claramente a qué parámetro se refiere (por ejemplo podría ser el número de orden de mantenimiento).
- La propiedad “nbits” indica la cantidad de bits necesarios para codificar el rango posible de variación del parámetro.
- El constructor nos permite crear una instancia del descriptor.
- El procedimiento “codificar_ADN”, recibe como parámetro una cadena de ADN y el offset al casillero en el que comienza la codificación del Genotipo. El parámetro “mask” es una máscara binaria de 16 bits, todos en cero, salvo en la posición del primer bit del Genotipo en la posición ADN[offset]. Al codificar el genotipo, se copia la representación binaria del parámetro Genotipo, a partir del bit determinado por “mask” dentro de ADN[offset] y se devuelven los parámetros “offset” y “mask” de forma de dejarlos apuntando al bit siguiente al último usado, dentro de la cadena para codificar el Genotipo. Fijando inicialmente offset=0, mask = 1, y llamando al procedimiento codificar_ADN sobre el vector de todos los

Genotipos, se obtiene en ADN la codificación total del individuo.

- La función “decodificar_ADN”, permite leer el genotipo a partir de una cadena de ADN. Los significados de los parámetros “offset” y “mask” son los mismos que los explicados en el párrafo anterior y permiten leer de la cadena de bits, los correspondientes al Genotipo particular y modificar los parámetros para prepararlos para que el próximo descriptor de genotipo pueda hacer su trabajo. El resultado de la función es TRUE si la decodificación binaria no precisó ser ajustada para cubrir el rango especificado para el parámetro y FALSE si hubo que hacer un ajuste. Para entender mejor el sentido de TRUE o FALSE como resultado, leer la descripción del manejo de Genotipos Enteros a continuación.

Como se puede apreciar, en esta clase genérica, nada se dice sobre el tipo de dato del parámetro “Genotipo”, solo se supone que en el lugar de memoria apuntado por dicho parámetro se pueden leer o escribir nbits.

Se definen clases refinadas de TDescriptorGenotipo, para facilitar el manejo de parámetros Booleanos, Reales y Enteros.

TDescriptorGenotipoBooleano. En este caso simplemente usa la clase genérica fijando nbits=1.

TdescriptorGenotipoEntero. En esta clase se agregan los parámetros “k_min y k_max” que deben ser pasados en el constructor y que fijan el rango admitido para el parámetro. La cantidad de bits se calcula en el constructor y se fija de forma tal que nbits sea el menor entero tal que (k_max-kmin) <= 2^nbits. Supongamos que se trata de un parámetro entero llamado “Camino” que puede tomar los valores 1, 2 o 3. Entonces, al crear la instancia del descriptor de genotipo llamaríamos al constructor así: camino:= TdescripotrGenotipoEntero.Create('Camino', 1, 3). Como consecuencia de dicha llamada, se creará un descriptor, en el que nbits = 2. Esto lleva a que en las cadenas de ADN existan 2 bits que están para representar este genotipo. Como 2 bits pueden representar 4 valores y sólo se necesitan 3 en este caso, hay más de una codificación binaria que terminará dando el mismo genotipo. Es aquí donde interviene el resultado de la función “decodificar_genotipo”. En la implementación se decidió realizar directamente la decodificación binaria del “tramo de bits” del ADN a un número entero obteniendo un número entre 2^nbits-1. El valor así obtenido se suma a k_min para obtener el valor del genotipo, si el resultado es mayor que k_max, se ajusta a k_max y se retorna FALSE para indicar que hubo que hacer un ajuste. Si el resultado es TRUE es que no fue necesario hacer ningún ajuste de rango.

TDescriptorGenotipoReal. Esta clase refinada de TdescriptorGenotipo es útil para el manejo de parámetros Reales (punto flotante). El constructor permite pasar como parámetros, el nombre del genotipo, el valor mínimo y máximo del parámetro y un parámetro “nbits” que determina la fineza con que se discretizará el rango (x_max – x_min) en la representación. El rango (x_max – x_min) será representado por 2^nbits puntos siendo por tanto la distancia entre los puntos de la discretización dx = (x_max – x_min) / (2^nbits -1). Como se puede apreciar, el parámetro real es representado por una discretización numerable, por lo que en esencia es tratado como si fuera un parámetro entero.

3.2. Definición de Clases básicas para descripción del problemas OddFace.

La unidad Pascal “uoddface” contiene las definiciones de las Clases Pascal útiles para la definición de problemas OddFace.

Las clases básicas son:

- TIndividuo. Esta Clase define la base de la cual derivar clases que puedan describir los individuos de cada problema concreto. Un individuo se diferencia de otro en esencia por su Genotipo, representado en su cadena de ADN. El Individuo está asociado a un Problema y es en el marco de ese problema que será evaluado y dependiendo de su desempeño será clasificado como más o menos apto.
- TProblema. Esta Clase define la base de la cual derivar clases para problemas concretos. El Problema contiene la descripción que permite poder evaluar el desempeño de los individuos asociados al problema. Por ejemplo, en un problema PIG (Planificación de Inversiones de Generación) el Problema contiene la información como para generar la Sala SimSEE a partir de un plan de inversiones dado (es decir un individuo) y para evaluar el costo resultante de ese plan de inversiones.
- TExplorador. Esta Clase generaliza el mecanismo de búsqueda de individuos. Como método principal tiene el de “proponer un nuevo individuo”. De esta clase básica es posible derivar exploradores refinados con diferentes estrategias para proponer nuevos individuos.
- TExploradorGenetico. Esta clase es un refinamiento de TExplorador. El mecanismo para proponer un nuevo individuo es en esta clase la del Cruzamiento de dos individuos seleccionados del conjunto de individuos ya evaluados del problema de acuerdo a un índice de performance para obtener así el ADN del nuevo individuo. El ADN resultante del cruzamiento puede a su vez sufrir cambios por la operación de Mutación.
- TExploradorMejorador. Esta clase es un refinamiento de TExplorador. El mecanismo para proponer un nuevo individuo es seleccionar uno de los mejores ya evaluados y simplemente repetir la evaluación con “otra suerte” para mejorar así la evaluación del individuo.

La solución de un problema pasa por activar exploradores que vayan buscando los mejores individuos (aquellos que minimizan la función objetivo de la optimización).

a) TIndividuo.

La clase TIndividuo tiene las propiedades:

- *Problema: TProblema;* Esta variable almacena una referencia al problema al que pertenece el individuo.
- *tipo_COD: integer;* Esta propiedad determina el tipo de codificación en que se almacena la representación binaria para las operaciones de Cruzamiento y Mutación. Los valores pueden ser: 0, 1 o 2 según que la codificación sea: BINARY, GRAY o UNARY.
- *nid: integer;* Ese es un identificador único del Individuo en la tabla de individuos evaluados asociados al problema al que pertenece el individuo. Si el valor es -1 (menos uno) significa que aún no se ha asignado un identificador único al individuo. Los identificadores únicos

deben ser solicitados al servidor de base de datos.

- *XR: TVectR;* Vector de genotipos reales. Contiene el conjunto de valores reales que representan los parámetros reales del individuo.
- *XE: TVectE;* Vector de genotipos enteros. Contiene el conjunto de valores enteros que representan los parámetros enteros del individuo.
- *ADN: TCadenaADN;* Representación del Genotipo (*XR* y *XE*) por una cadena de bits.
- *f_VE, f_VaR, f_CVaR, f_MIN, f_MAX, f_objetivo: NReal;* Estas variables almacenan las estimaciones realizadas en base a conjuntos de crónicas (muestreos de Monte Carlo) del valor esperado de la función de costo, del que es excedido con probabilidad 5%, del valor esperado del conjunto del 5% más alto de costos, el valor mínimo del conjunto muestreado, el valor máximo del conjunto muestreado y el valor de la función objetivo que se busca minimizar respectivamente. El valor objetivo se compone como una combinación lineal de *f_VE, f_VaR y f_CVaR* según los pesos especificados en la definición del problema. Estos estimadores son obtenidos para cada evaluación (realizada con un número de muestreos de Monte Carlo) y si el individuo es evaluado más de una vez los promedios de los estimadores obtenidos en cada evaluación son almacenados en las variables *f_VE, f_VaR, f_CVaR, f_objetivo*, y en las variables *f_MIN, f_MAX* se almacenan el mínimo y el máximo de los valores estimados en cada evaluación.
- *cnt_evaluaciones: integer;* Cantidad de veces que fue evaluado este individuo. Cada evaluación consiste en la realización de un número de muestreos de la función de costo definido en la especificación del problema. El mismo individuo pudo haber sido evaluado más de una vez y el valor *cnt_evaluaciones* da cuenta de eso.
- *f_histo: TVectR;* Esta propiedad es del tipo vector de reales y almacena directamente el valor de la función de costo obtenida por cada simulación de Monte Carlo realizada durante una evaluación. Por ejemplo, si en la especificación del problema se estableció que cada evaluación se realice sobre 100 crónicas (muestreos) entonces el vector *f_histo* tendrá los 100 valores obtenidos durante la simulación. En el caso en que un individuo haya sido evaluado más de una vez, el vector *f_histo* almacena el promedio de los valores *f_histo* que tendría cada evaluación en forma independiente si fuera la única.

```
// crea un nuevo individio "limpio".
constructor CreateNew(problema_: TProblema);

// crea un individuo con un ADN dado poniendo a cero todos los demás parámetros.
// es útil para decodificar la cadena.
constructor CreateFromADN_HexStr( problema_: TProblema; ADN_HexStr:string );

// crea un individo desde un record de la DB
constructor CreateFromRec(problema_: TProblema; r: TDataRecord);

// comunica el resultado al la DB
```

```
    procedure ComunicarResultado;

    // convierte el ADN a codificacion GRAY
    procedure toGray;

    // convierte el ADN a codificación BINARY
    procedure toBinary;


    function ADN_AsBinaryStr: string;
    procedure Free; virtual;
  end;
```

4. Consideraciones adicionales.

A continuación se describen dos aspectos importantes a considerar al momento de desarrollar un nuevo tipo de Explorador de problemas en OddFace.

4.1. Muestreos de Monte Carlo.

La primera observación es que, al tratarse de la búsqueda de un mínimo de una función que es evaluada mediante un conjunto de sorteos de Monte Carlo por simulación, y dado que la búsqueda del mínimo es en definitiva la comparación entre la evaluación en diferentes puntos del espacio de búsqueda, las técnicas de reducción de varianza por muestreo sincronizado valen para este caso. En otras palabras, sería deseable comparar los valores para el mismo conjunto de realizaciones de los procesos estocásticos, o por lo menos que, en lo posible, es decir en aquellos aspectos que los procesos estocásticos involucrados no dependen del punto de evaluación, se mantenga para ambas evaluaciones.

A modo de ejemplo, en el caso del PAM, la rotura fortuita de las máquinas debiera mantenerse incambiada al igual que otros procesos como ser las velocidades de viento o los caudales de aportes a las centrales hidroeléctricas, para que los mismos no agreguen diferencias a la comparación.

Para lograr esta mejora, bastaría con introducir un orden en las semillas aleatorias con que se evalúan los puntos.

Si el problema se resolviera admitiendo una sola evaluación de NCronicas para cada individuo, resulta evidente que la mejor solución sería usar la misma semilla aleatoria para todas las evaluaciones. Tal como está implementada la generación de números aleatorios en SimSEE, a partir de la versión 2.63 “Anarquía” (la semilla aleatoria) determina la generación de números aleatorios de cada componente de la simulación en forma independiente, con lo cual, al evaluar dos individuos, que en definitiva significa realizar la optimización y simulación de dos Salas SimSEE, si se utiliza la misma semilla aleatoria, la generación de aleatoriedad en cada componente que permanezca incambiado en ambas Salas será idéntica.

En el manejo de las realizaciones que se utilizan durante la simulación hay que asegurar que, para el

mismo número de evaluaciones, los Individuos fueron testeados sobre los mismos conjuntos de realizaciones en aquello que se pueda mantener, aunque los individuos sean diferentes. En las implementaciones de OddFace_PIG, OddFace_PAM y OddFace_OptimA est se logra imponiendo que las semillas de simulación son calculadas a partir de una Semilla_Madre (igual para todos) más el número de evaluación. Generada así, una semilla dependiente del número de evaluación, de mantener la coherencia interna se encarga SimSEE.

4.2. Zonas de Indiferencia en la cadena de ADN.

Al seleccionar los rangos posibles de los parámetros y cómo los mismos afectan el problema bajo evaluación, hay que tener la precaución de no dejar Zonas de Indiferencia. Por Zonas de Indiferencia nos referimos a regiones del Dominio que al ser traducidas al Problema (por ej. al crear la Sala SimSEE en el caso de OddFace_PIG) sean Individuos, a todos los efectos, iguales. Las Zonas de Indiferencia enlentecen la convergencia del algoritmo de Exploración genético. A nivel de las cadenas de ADN los individuos son diferentes, pero a nivel del problema a resolver son idénticos y se termina creando cruzamientos sobre Individuos que no generan exploraciones realmente diferentes.

4.3. Individuos infactibles y residuo de bits en la cadena de ADN.

En la implementación de OddFace, al proponer un nuevo Individuo, podría ocurrir que el mismo sea Infactible, indicando con eso que no es posible realizar la simulación correspondiente para evaluar el costo del Individuo. Se debe implementar en el propio Explorador, una función que, dado un Individuo, realice los chequeos necesarios y determine si el mismo es o no factible. El que exista la posibilidad de generar Individuos Infactibles generalmente indica que la forma de codificar los parámetros del Problema podría ser mejorada, evitando así la generación de Individuos Infactibles. Solo a modo de ejemplo, supóngase un problema con dos parámetros $x_1 \in [0,1]$ y $x_2 \in [0,1]$ pero que para que el Individuo sea factible se debe cumplir $x_1 + x_2 < 1$. Si se piensa que se dejará proponer individuos variando libremente x_1 y x_2 se tendrá que el 50% de la región a explorar corresponde a Individuos inviables. En este ejemplo sería preferible hacer un cambio de variable y en lugar de x_2 considerar como parámetro $r \in [0,1]$ y poner dentro de la codificación del problema el cálculo $x_2 = x_1 r$, evitando así regiones con individuos infactibles.

5. Anexo. Probabilidad de Mutación.

En el formulario de edición del problema (ver Fig.7) es posible definir la probabilidad de mutación. Este parámetro se utiliza para construir un generador de números aleatorios con la distribución

$$p_N(m) = C_m^N P_{Mut}^m (1 - P_{Mut})^{(N-m)} \qquad \text{ec.(2).}$$

donde N es el largo de la cadena de ADN del problema (cantidad de bits) y P_{Mut} es la probabilidad de mutación especificada en el formulario. Con la densidad de probabilidad de la ec.2 se genera el número m de mutaciones a realizar. Estas mutaciones se realizan al azar sobre la cadena de bits en forma secuencial, por lo cual un bit invertido en una de las mutaciones puede volverse a invertir en las siguientes.

Dada una secuencia de N bits, si en el paso 1 se selecciona cualquiera de los bits (con igual

probabilidad) y se invierte, la probabilidad de invertir un bit dado es $p_1=\frac{1}{N}$. Si en el paso 2 se vuelve a seleccionar cualquiera de los bits y se invierte, la probabilidad de haber invertido un bit dado en la sucesión de pasos es $p_2=p_1(1-\frac{1}{N})+(1-p_1)\frac{1}{N}$

En forma similar, se puede escribir la probabilidad de que al final de paso $k+1$ un bit dado resulte invertido en la sucesión de pasos como:

$$p_{k+1}=p_k(1-\frac{1}{N})+(1-p_k)\frac{1}{N}=p_k(1-\frac{2}{N})+\frac{1}{N}$$

$$p_{k+1}=p_k(1-\frac{2}{N})+\frac{1}{N} \qquad \text{ec.(3)}$$

La ec.3 tiene como punto unido $p_{k+1}=p_k=\frac{1}{2}$ por lo que haciendo el cambio de variable $z_k=p_k-\frac{1}{2}$ se tiene la ecuación

$$z_{k+1}=z_k(1-\frac{2}{N}) \qquad \text{ec.(4).}$$

La ec.4 tiene solución $z_k=(1-\frac{2}{N})^k z_0$ con $z_0=(p_0-\frac{1}{2})=-\frac{1}{2}$

De lo anterior se puede escribir la probabilidad de que luego de m pasos un bit dado resulte invertido (o sea que haya sido invertido un número impar de veces) como:

$$p_m=\frac{1}{2}-(1-\frac{2}{N})^m\frac{1}{2}=\frac{1}{2}(1-(1-\frac{2}{N})^m) \qquad \text{ec.(5)}$$

Los valores posibles de m son entre 0 (cero) y N . La Fig.13 muestra que ec.5 es visualmente, cuasi-independiente de N cuando se grafica contra m/N

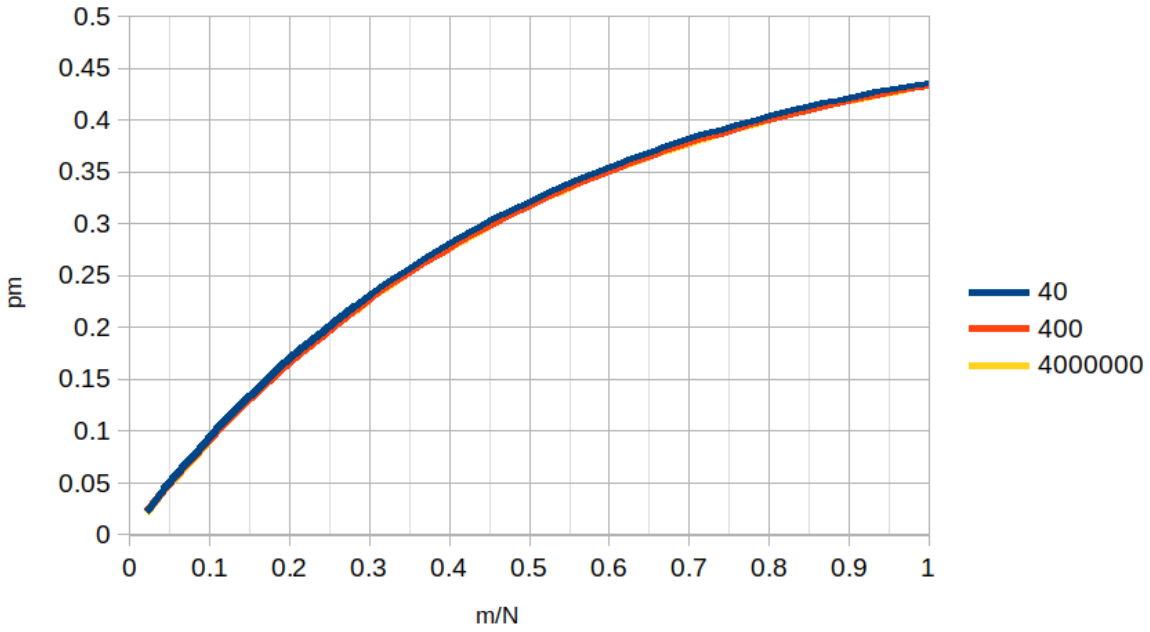

Fig. 13: Probabilidad de mutación de cada bit en una secuencia de m mutaciones.

Entonces, el valor esperado de los bits invertidos al aplicar m mutaciones secuenciales será:

$$Np_m = \frac{N}{2}\left(1-\left(1-\frac{2}{N}\right)^m\right) \qquad \text{ec.(6).}$$

$$C_m^N = \frac{N!}{m!(N-m)!}$$

6. OddFace-PIG. Programación de Inversiones en Generación.

6.1. Introducción.

El uso de la herramienta OddFace se explica en el Manual de OddFace. Este documento tiene por objeto describir la aplicación OddFace-PIG que permite definir sobre OddFace el problema de optimización del plan de inversiones en generación.

Para evaluar un plan de inversiones en generación, el usuario debe suministrar una Sala SimSEE e indicar qué Actores de dicha Sala son aquellos sobre los que es posible instalar nuevas unidades. Para ello, se configuran las posibles tecnologías sobre las que es posible expandir (por ej. Eólica, Solar, Turbinas de ciclo abierto) y el horizonte temporal en el que es posible tomar decisiones así como la frecuencia con que es posible tomar las mismas.

Para definir un problema del tipo OddFace_PIG debe utilizar la herramienta OddFace_Prepare. En el Manual de Usuario de esa herramienta encontrará cómo configurar el acceso de Usuario y cómo iniciar la creación de un nuevo problema. Cuando inicie la creación de un problema se abrirá un formulario como se muestra en la Fig.14.

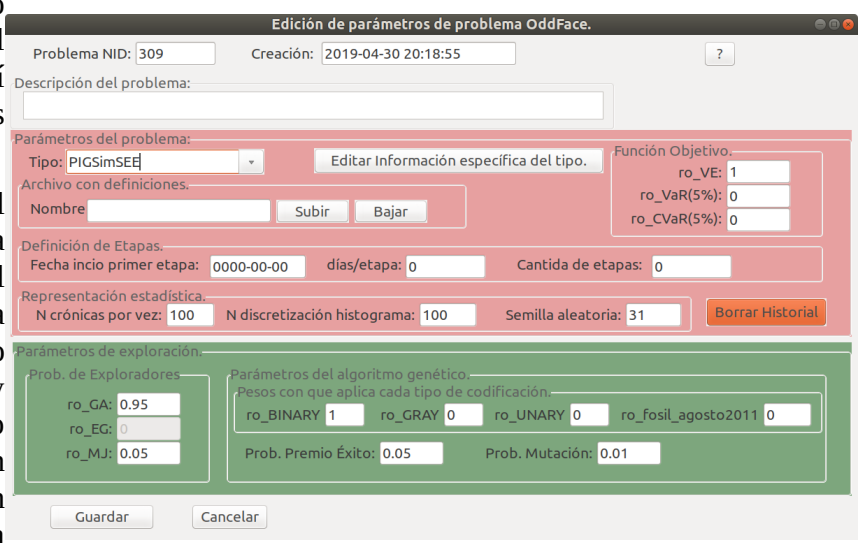

Fig. 14: Formulario de edición de un Problema OddFace.

6.2. Parámetros generales.

En este formulario de edición del problema (Fig.14), debe seleccionar en el panel “Parámetros del problema” en el selector “Tipo” el tipo de problema PIGSimSEE. Una vez seleccionado el tipo de problema con el botón “Subir” seleccione el archivo de Sala SimSEE (archivo con extensión “.ese”) y se iniciará la suba de la Sala al servidor.

Antes de subir la Sala, identifique los Actores sobre los que quiere realizar la expansión. Deberá ingresar el nombre de esos Actores como opciones de tecnologías en la edición de los parámetros específicos del problema. Debe asegurarse que estos actores tengan una sola Ficha Unidades con CERO unidades instaladas con fecha anterior al inicio de la simulación. Esto es importante pues si dejan fichas con unidades instaladas en los Actores que son opción de expansión, luego no podrá diferenciar entre las unidades instaladas por el optimizador y las que ya existían en la Sala.

6.3. Configuración de una tecnología.

Presionando el botón “Editar información específica del tipo.” en el formulario de edición del problema (Fig.14) se abre el editor de tecnologías que serán las opciones de inversión que se muestran en la Fig.15. Este listado se crea inicialmente vacío y mediante el botón “Agregar Nueva” se introducen las tecnologías que son opciones de inversión.

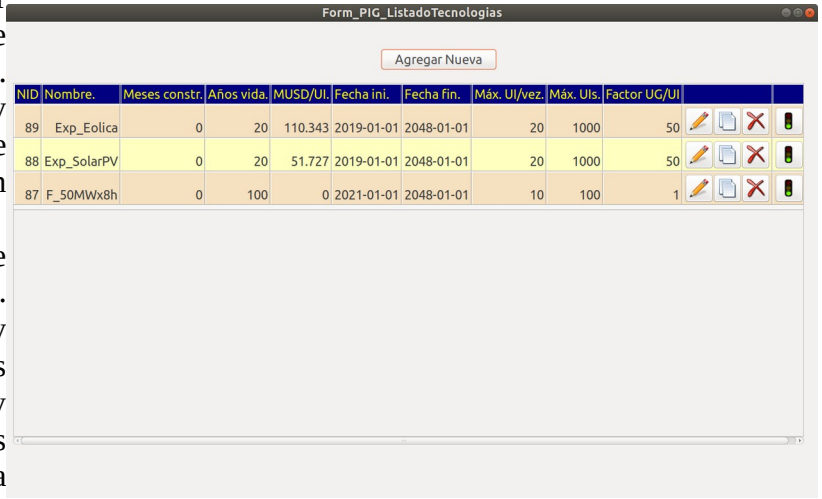

Fig. 15: Listado de las tecnologías opciones de inversión.

En la Fig.15 se muestra que se configuraron tres tecnologías. “Exp_Eolica”, “Exp_SolarPV” y “F_50MWx8h”. Esto son solo ejemplos para una Sala SimSEE en la que hay Actores con esos nombres previstos para expandir la generación en base a instalar unidades en los mismos.

En cada renglón del listado se dispone de una botonera:  . El “lápiz” le permite editar los parámetros de la tecnología, las “dos hojas” le permiten agregar una nueva tecnología al listado copiando los parámetros de una existente, la “cruz” le permite eliminar una tecnología y el “semáforo” le permite activar o desactivar la tecnología (solo las activas son consideradas en la optimización).

Tenga en cuenta que si cambia cualquier parámetro de una tecnología o activa o desactiva o elimina o agrega tecnologías estará cambiando al estructura de El Problema y por tanto si el mismo ya estaba en ejecución deberá eliminar las evaluaciones ya realizas utilizando el botón “Borrar Historial” del formulario de edición del problema Fig.14.

Presionando el botón “Agregar Nueva” en el listado de tecnologías (Fig.15) o clonando una existente con el botón  se abrirá el formulario de edición de una tecnología que se muestra en la Fig.16.

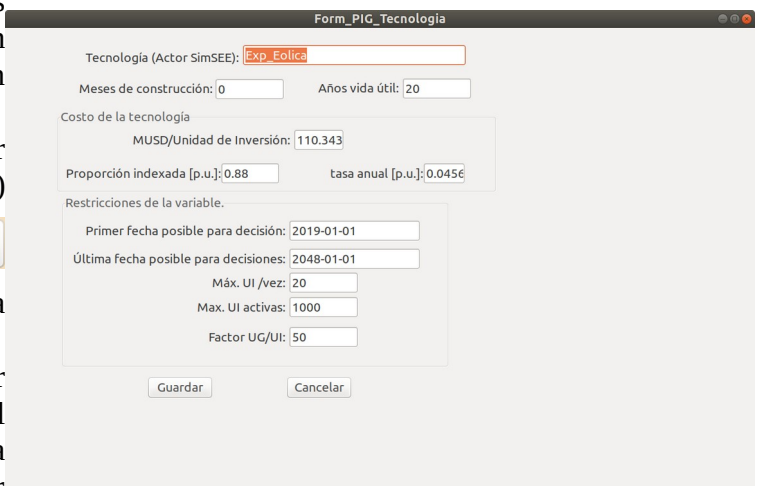

Fig. 16: Formulario de edición de una tecnología.

En el campo “Tecnología (Actor SimSEE):” se debe ingresar el nombre del Actor de SimSEE que se utilizará para expandir la generación en base a instalar Unidades de generación agregando las fichas correspondientes en el Actor.

El campo “Meses de construcción” permite definir los meses que se estima pasarán desde la decisión de inversión hasta que los proyectos del tipo de tecnología en cuestión entran en operación. La utilización de este campo es opcional. Si especifica un valor 0 (Cero) simplemente estará representando que las unidades ingresan en el mismo momento en que se toma la decisión. Si especifica un valor superior a cero, debe tener en cuenta que si la inversión se configura como se especifica en la sección 6.3.b , entonces la inversión es realizada en una fecha y la nueva generación ingresará luego de transcurrida la cantidad de meses especificada.

El campo “Años de vida útil” le permite especificar cuál es la vida útil de una central de

generación de la tecnología en cuestión. En la Fig.16 se especificó 20 años, lo que implica que cuando OddFace agrega una ficha con unidades de la tecnología en la Sala en una fecha dada, en la misma fecha 20 años después modifica la ficha de unidades para representar el retiro de las unidades del sistema.

El Panel “Costo de la tecnología” permite especificar los costos fijos asociados a la instalación de Unidades de Inversión (UI) de la tecnología y se explica detalladamente en la sección 6.3.b

En el Panel “Restricciones de la variable” debe acotar la variable de decisión (la cantidad de unidades a instalar).

Debe especificar la “Primera fecha posible para la decisión” y la “Última fecha posible para decisiones”. Estos dos campos actúan como un filtro temporal adicional a la definición de las Etapas de decisión y solamente pueden restringir el horizonte de tiempo definido por las Etapas.

En el campo “Máx. UI/vez” debe especificar la máxima cantidad de unidades de la tecnología que es razonable se pueda instalar en una etapa de decisión. Por ej. si el crecimiento de la Demanda del sistema es de 100 MW por Etapa de decisión y una UI representa un Parque Eólica de 50 MW los incrementos de Demanda serán cubiertos con 4 parques por etapa (suponiendo un factor de capacidad de 0.4) y suponiendo que en la misma etapa se pueden retirar parques anteriores un valor justo podría ser 8 y para tener algo de holgura se podría configurar “Max. UI/vez = 10”.

El campo “Máx. UI activas” especifica la cantidad máxima de unidades de inversión activas que es razonable en el sistema. A modo de ejemplo, si la Demanda máxima es de 2000 MW y si una UI fuera un parque eólico de 50 MW con factor de capacidad 0.4 no tendría sentido mantener activos mucho más que 2000 / (50*0.4) = 100 Unidades de Inversión.

Es importante reducir en lo posible los rangos de las variables, dado que a mayor rango, mayor dificultad tendrá el algoritmo de optimización para explorar el dominio de soluciones posibles y por tanto mayor será el tiempo de cálculo.

El parámetro “UG/UI” determina cuántas Unidades de Generación se incorporan en el Actor correspondiente de la Sala SimSEE por cada Unidad de Inversión. En el ejemplo de la Fig.16 el valor de 50 implica que por cada Unidad de Inversión de OddFace, en el Actor “Exp_Eolica” de la Sala del ejemplo se instalarán 50 nuevas unidades de generación. Como en dicha Sala el Actor Exp_Eolica está configurado con unidades de 1 MW, una UI corresponde a 50 MW instalados en la Sala.

a) Costos fijos como pagos por disponibilidad en la Sala.

Los Actores dentro de la Sala SimSEE disponen de un campo (en su formulario principal o en las fichas de parámetros dinámicos) que permite especificar un pago por disponibilidad en USD/MWh (dólares por MW y por hora disponible). Este pago por disponibilidad (a veces llamado Pago por Potencia) aparece en la Sala como un costo. Al agregar unidades al Actor, automáticamente se está considerando el aumento de los costos fijos del sistema por aplicarse este pago por disponibilidad.

Esta forma de considerar los costos fijos es sencilla, pero dificulta la consideración de costos de inversión con decaimiento en el tiempo. Para poder considerar este tipo de costos con decaimiento se debe usar la opción de configuración que se explica en la sección siguiente 6.3.b.

b) Costos fijos como inversiones puntuales en OddFace-PIG.

En la sección 6.3.a se especificó una forma de considerar los costos fijos (Inversión + O&M) en los Actores de una Sala mediante el Pago por Disponibilidad. En esta sección se muestra un mecanismo alternativo, que permite representar en forma simplificada el decaimiento en el tiempo de los costos de inversión que generalmente están asociados a las mejoras en la producción de las tecnologías.

El Panel “Costo de la tecnología” (ver Fig.16) permite establecer el costo inicial de inversión por cada Unidad de Inversión (UI) de la tecnología y su evolución en el tiempo.

El parámetro “MUSD/Unidad de Inversión” debe completarse con los millones de dólares que cuesta una Unidad de Inversión de la tecnología en el instante t_0 .

El parámetro “Proporción indexada [p.u.]” debe especificar qué proporción del valor especificado en el parámetro anterior está sujeto a un decaimiento en el tiempo. En el ejemplo de la Fig.16, el valor 0.88 está indicando que en este caso se considera que el 88% del monto de la inversión por unidad está sujeto a un decaimiento en el tiempo e indirectamente que el 12% restante permanece con valor constante.

El parámetro “tasa anual [p.u.]:” debe contener la tasa anual con que decae la porción indexada. La ec.7 muestra cómo se calcula el costo de la Unidad de Inversión de la tecnología C_t en el instante t .

$$C_t = C_{t_0}\left[f_V\left(\frac{1}{1+\alpha}\right)^{t-t_0} + (1-f_V)\right] \qquad \text{ec.(7) indexación del costo.}$$

donde el instante t_0 es el Inicio de la simulación o la fecha de Guarda de la Simulación (lo que resulte posterior). Ambos valores son los especificados en la Sala SimSEE utilizada. El valor C_{t_0} es el valor presente de los costos fijos al instante t_0 . El factor f_V corresponde a la porción del costo que decae anualmente a la tasa de decaimiento α . En la ec.7 tanto t como t_0 están expresados en años.