Scientia Et Technica
ISSN: 0122-1701
scientia@utp.edu.co
Universidad Tecnológica de Pereira
Colombia

Arboleda Molina, Orlando; Sotelo, Steven
Construcción de aplicativos de programación por restricciones en Microsoft Solver Foundation y Windows Azure
Scientia Et Technica, vol. 21, núm. 4, diciembre, 2016, pp. 336-341
Universidad Tecnológica de Pereira
Pereira, Colombia

Construcción de aplicativos de programación por restricciones en Microsoft Solver Foundation y Windows Azure

Construction of constraints programing applications in Microsoft Solver Foundation and Windows Azure

Orlando Arboleda Molina, Steven Sotelo
Departamento de Operaciones y Sistemas, Universidad Autónoma de Occidente, Cali, Colombia
oarboleda@uao.edu.co
stevenbetancurt@hotmail.com

***Resumen*— A nivel empresarial y académico se requiere la construcción de aplicativos para resolver problemas de optimización de restricciones, y que estos puedan ser ofrecidos como servicios en la nube. En el presente artículo se modela y resuelve un problema de optimización de restricciones, correspondiente al problema de transporte, usando para su modelamiento y posterior publicación como servicio en la nube, solo tecnología .Net, aplicable a problemas computacionalmente complejos de restricciones, usando Microsoft Solver Foundation que permite la construcción de poderosos aplicativos genéricos de pocas líneas de código, los cuales pueden ser fácilmente publicados como servicios web en Windows Azure.**

***Palabras clave*— *Problemas de optimización de restricciones, Microsoft Solver Foundation, Windows Azure, computación en la nube, servicios web, modelo general de transporte*.**

***Abstract*— At business and academic level is required the construction of applications to solve constraints optimization problems, and that they could be offered as services in the cloud. In the present article is modeling and solve one constraint optimization problem, corresponding to the transport problem, using for his modeling and later publication as service in the cloud, only .Net technology, applicable to computationally complex constraint problems, using Microsoft Solver Foundation that allows the construction of powerful generic applications of few lines of code, which can be easily published as web services in Windows Azure.**

***Key Word* — constraints optimization problems, Microsoft Solver Foundation, Windows Azure, cloud computing, web services, general transportation model.**

I. INTRODUCCIÓN

A nivel empresarial y académico es de vital importancia el contar con aplicativos que permitan resolver problemas de planificación y optimización de restricciones, y que estos puedan ser ofrecidos como servicios en la nube. Por ello, es necesario explorar herramientas que permitan un fácil modelado de los sistemas, que sean eficientes en la obtención de las soluciones y que permitan fácilmente proveer los aplicativos desarrollados como servicios en la nube.

Teniendo en cuenta, que muchos desarrolladores hacen uso de tecnologías .Net, y que estas presentan novedades constantes en sus herramientas y lenguajes de programación, es necesario explorar sus apuestas para abordar problemas de optimización de restricciones, y para la publicación de soluciones en la nube.

En el presente artículo se muestra una implementación de un problema de optimización de restricciones para el modelo general del transporte, usando solo tecnologías .Net, describiendo las potentes estructuras encontradas en Microsoft Solver Foundation que permitieron la construcción de una solución de pocas líneas de código y adaptable a modelos de mayor envergadura; y como fácilmente se logra proporcionar el desarrollo obtenido como un servicio en la nube a través del Windows Azure.

II. PROGRAMACION POR RESTRICCIONES

La programación por restricciones es un modo alterno de programación, en el cual el proceso de programación se limita

Fecha de Recepción: 26 de noviembre de 2015
Fecha de Aceptación: 20 de diciembre de 2016

a la generación de requerimientos o restricciones y a la obtención de una solución a través de métodos generales los cuales combinan técnicas de reducción del espacio de búsqueda con métodos de búsqueda específicos o métodos de dominio especifico que corresponden a algoritmos de propósito especial o paquetes especializados, por ejemplo para la resolución de ecuaciones lineales o paquetes para programación lineal [1].

La programación por restricciones también consiste en un conjunto de técnicas para resolver problemas de satisfacción de restricciones (Constraint Satisfaction Problems o CSP), los cuales consisten en una secuencia finita de variables con sus respectivos dominios y un conjunto de restricciones. Aunque un CSP siempre puede ser resuelto con búsqueda de fuerza bruta en problemas muy pequeños, en otros debe ser complementada con procesos de reducción de la cantidad de búsqueda necesaria [2].

Existen problemas de optimización de restricciones (Constrained Optimization Problems o COP), los cuales constan de un CSP y una función objetivo que debe ser optimizada (maximizada o minimizada). El propósito en este caso es encontrar la solución óptima.

Muchos de los problemas reales a resolver pueden ser modelados como problemas de satisfacción u optimización de restricciones y estos tradicionalmente suelen ser modelados en lenguajes de programación como Mozart o con librerías de gran desempeño como Gecode, los cuales son resueltos con sistemas o motores genéricos de programación por restricciones, en los cuales se ha de combinar procesos de propagación o eliminación de valores no permitidos en las soluciones con procesos de búsqueda o aplicación de heurísticas, los cuales permiten ser aplicados a una amplia gama de problemas, pero sus tiempos de respuesta son inferiores a los producidos por sistemas específicos, para los cuales ya existen algoritmos completos y eficientes [1] y[2].

III. MICROSOFT SOLVER FOUNDATION

Microsoft Solver Foundation (MSF) es un sistema basado en .Net desarrollado para contribuir a una plataforma integrada de planificación y optimización de negocios. Permite planeación, planificación y optimización. Puede ejecutarse de forma independiente (por ejemplo, como un complemento para Excel 2010) o como parte de un sistema de planeación/planificación con restricciones de tiempo real, permitiendo el modelamiento de alto nivel a través de lenguajes .Net ampliamente difundidos como C# y C++.

MSF es un ambiente de modelamiento que se enfoca en los factores claves y resultados de las decisiones, que puede ser aplicado a problemas computacionalmente complejos, permitiendo trabajar con modelos de programación: lineal, entera, de dominio finito, no lineal, estocásticos y dinámicos[3], aplicable a campos tan diversos como: optimización en tiempo real de cadenas de suministro, portafolios de investigación de análisis de riesgo, investigación de operaciones, modelamiento de riesgos, optimización de decisiones, etc [4].

MSF dentro de su suite de herramientas ofrece un API programable llamado Solver Foundation Services (SFS) constituida por un conjunto de clases dentro del framework .Net. Este conjunto de servicios son para el modelado y solución de problemas de distintas índole. Para modelar y solucionar un problema con SFS es necesario comprender las principales clases que ofrece y el flujo general que ha de seguirse [5]:

- *SolverContext:* es el punto de entrada básico del modelo. Este indica el hilo de ejecución que mantendrá el actual modelo, el cual provee un enlace con los datos del mismo.
- *Modelo (model):* esta clase representa el modelo a solucionar. Contiene los parámetros, variables de decisión, restricciones y metas en formato algebraico
- *Dominio (domain):* define el conjunto de posibles valores que puede tomar un variable o parámetro dentro del modelo. Puede ser entero, booleano, real o una enumeración.
- *Decision (decision):* son las variables que representan las salidas del modelo planteado.
- *Parametro (parameter):* representan las entradas del modelo.
- *Restriccion (constraint):* son expresiones algebraicas que representan la relación que deben tener las variables de decision y los parametros.
- *Funcion objetivo (goal):* expresión algebraica que indica la función de optimización de un modelo COP, la cual puede ser de maximización o minimización.
- *Conjunto (set):* con esta clase podemos representar un conjunto de los posibles valores claves de las decisiones o los parametros. Es una buena alternativa para modelos que contienen varios valores.
- *Solucion (solution):* la solución del modelo planteado. Con esta clase se puede solucionar el modelo y obtener los valores de las variables de decisión.
- *Reporte (report):* ofrece una descripción detalla de los resultados obtenidos en la solución del modelo.
- *Directiva (directive):* usada para indicar que algoritmo o solver usar para la solución del modelo.

Para obtener soluciones en MSF es necesario seguir el flujo general indicado en la Figura 1

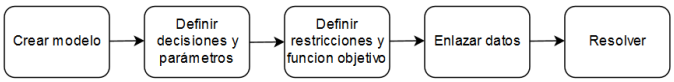

Figura 1. Flujo general para resolver un modelo con MSF

Al igual que en otros lenguajes o liberarías para la resolución de modelos de restricciones, en MSF el programador debe indicar el algoritmo o solver que el sistema ha de usar para computar las soluciones y su escogencia puede incrementar o reducir el tiempo de ejecución necesario para encontrar las soluciones. En MSF el solver se impone a través de la aplicación de una directiva, que puede ser *ConstraintProgrammingDirective* si se desea un solver de propósito general o *SimplexDirective* si se desea el solver de propósito específico para resolución de modelos lineales que tendrá mejor desempeño.

IV. WINDOWS AZURE

Los servicios Windows Azure es el producto que ofrece Microsoft en la nube. Los servicios de Cloud Computing ofrecidos en esta plataforma van desde infraestructura como servicio (IaaS), mediante máquinas virtuales con sistemas operativos de servidor Windows y Linux; plataforma como servicios (SaaS), en una amplia gama de productos comerciales; servicio de aplicaciones, con la cual se pueden desarrollar aplicaciones propias y subirlas a la nube.

Por su simplicidad y escalabilidad en el desarrollo, despliegue y administración de las aplicaciones y servicios ofrecidos en Windows Azure, es el servicio más indicado para realizar la implementación de software que requiera gran cantidad de recursos [6].

Windows Azure es una buena alternativa para la solución de modelos de programación por restricciones, los cuales tienden a demandar muchos recursos y estos pueden depender de la cantidad de variables y parámetros.

V. MODELO IMPLEMENTADO EN MSF

A. *Modelo a implementar*

El modelo a implementar corresponde a un caso particular del problema de Transporte, en el cual desde 3 fábricas se envían productos a 5 distribuidores, con las disponibilidades, requerimientos y costos unitarios de transportes indicados en la Figura 2. En este caso se desea determinar qué cantidad del producto se debe enviar desde cada fábrica a cada distribuidor para minimizar los costos del transporte [7].

Fábricas	Distribuidores					Disponibilidades
	1	2	3	4	5	
1	20	19	14	21	16	40
2	15	20	13	19	16	60
3	18	15	18	20	X	70
Requerimientos	30	40	50	40	60	

Figura 2. Modelo a implementar

Este problema de transporte debe ser modelado con el sistema de restricciones mostrado en la Figura 3.

B. *Implementacion realizada en MSF*

Para la implementación del modelo de transporte propuesto, se usó la versión 3.1 de MSF, en un equipo de escritorio con la siguiente configuración: Procesador Intel Core I5 de 4 núcleos, con 4Gb DDR3 de memoria Ram, disco duro de 700Gb, corriendo como sistema operativo Windows 8.1, entorno de desarrollo Visual Studio 2013 y Framework .Net 4.

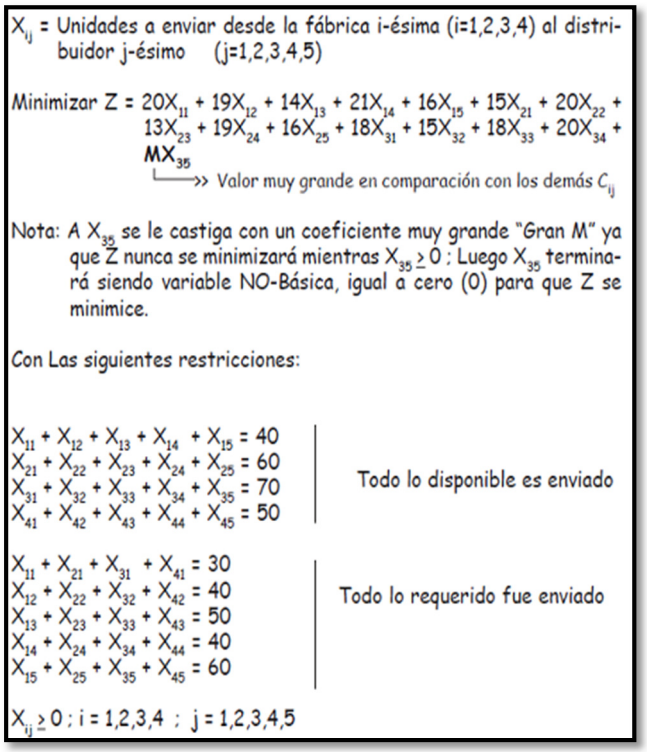

Figura 3. Sistema de restricciones a implementar

La solución se implementó como un Web Service utilizando el framework de Windows Communication Foundation (WCF)1. La entrada corresponde a una cadena de texto formateada en XML existente en el archivo Parametros.xml, con la estructura indicada en la Figura 4

[1] presente en https://github.com/stcuao/msf_modelo_transporte

```
<?xml version="1.0" standalone="true"?>
<parametros>
  - <demanda>
        <distribuidor>Distribuidor 1</distribuidor>
        <valor>30</valor>
    </demanda>
  - <demanda>
        <distribuidor>Distribuidor 2</distribuidor>
        <valor>40</valor>
    </demanda>
  - <costo>
        <fabrica>Fabrica 1</fabrica>
        <distribuidor>Distribuidor 1</distribuidor>
        <valor>20</valor>
    </costo>
  - <costo>
        <fabrica>Fabrica 1</fabrica>
        <distribuidor>Distribuidor 2</distribuidor>
        <valor>19</valor>
    </costo>
  - <disponibilidad>
        <fabrica>Fabrica 1</fabrica>
        <valor>40</valor>
    </disponibilidad>
  - <disponibilidad>
        <fabrica>Fabrica 2</fabrica>
        <valor>60</valor>
    </disponibilidad>
  - <disponibilidad>
        <fabrica>Fabrica 3</fabrica>
        <valor>70</valor>
    </disponibilidad>
</parametros>
```

Figura 4. Fragmento del archivo Parametros.xml

Como se indica en la Figura 5, la entrada se cargó en una estructura Dataset para su fácil acceso a los datos al interior de la aplicación. Previamente se serializó la entrada en un objeto especializado para XML llamado XDocument.

Para enlazar los datos con el modelo es necesaria una estructura que implemente la interface IEnumerable de .Net, por esto se creó una instancia de la clase DataTable y se configuró para que sus columnas recibieran el tipo de datos de la entrada al servicio y luego se cargaron en ella los datos de entrada.

```
XDocument documento = XDocument.Parse(parametros);
DataSet ds = new DataSet();
ds.ReadXml(documento.CreateReader());

DataTable tDemanda = ds.Tables["demanda"].Clone();
tDemanda.Columns["valor"].DataType = typeof(int);
tDemanda.Columns["distribuidor"].DataType = typeof(string);
foreach (DataRow row in ds.Tables["demanda"].Rows)
    tDemanda.ImportRow(row);
```

Figura 5. Carga de los datos de entrada al modelo

Como todo modelo de restricciones fue necesario crear el store o contexto como suele denominarse en MSF sobre el cual definió el modelo. Se recomienda limpiar el contexto para evitar problemas con la memoria cache, como se indica en la Figura 6.

```
SolverContext context = SolverContext.GetContext();
context.ClearModel();
Model model = context.CreateModel();
```

Figura 6. Creación del contexto

Aunque el modelo propuesto cuenta con solo 20 variables, en un modelo real el número de fábricas y distribuidores puede ser muy grande, por ello se realizó una implementación que pudiera soportar cualquier cantidad de variables de decisión y restricciones. Para lograr dicha adaptabilidad a modelos mayores, se usaron conjuntos (set), los cuales permitieron almacenar las fábricas y los distribuidores, y su posterior enlazamiento dinámico (binding) con los datos de entrada al modelo, a través de parámetros.

Como se indica en la Figura 7, se crearon parámetros para las demandas, costos y disponibilidades, a los cuales se les configuro el dominio, nombre y conjunto con el que se enlazaron los datos. Posteriormente enlazamos a cada parámetro estructuras que implementan la interface IEnumerable de .Net, con los campos de entrada al sistema.

Con los parámetros enlazados al modelo, se definió en el modelo una única variable de decisión con los conjuntos que contienen la información de los parámetros. Esta será capaz de identificar la cantidad de variables necesarias, que se requieren como salida del modelo.

```
Set fabricas = new Set(Domain.Any, "fabricas");
Set distribuidores = new Set(Domain.Any, "distribuidores");

Parameter demanda = new Parameter(Domain.Integer, "demanda", distribuidores);
demanda.SetBinding(getDemanda().AsEnumerable(), "valor", "distribuidor");

Parameter costos = new Parameter(Domain.Integer, "costos", fabricas, distribuidores);
costos.SetBinding(getCostos().AsEnumerable(), "valor", "fabrica", "distribuidor");

Parameter disponibilidad = new Parameter(Domain.Integer, "disponibilidad", fabricas);
disponibilidad.SetBinding(getDisponibilidad().AsEnumerable(), "valor", "fabrica");

model.AddParameters(demanda, costos, disponibilidad);

Decision x = new Decision(Domain.RealNonnegative, "x", fabricas, distribuidores);
model.AddDecision(x);
```

Figura 7. Conjuntos y parámetros usados

Como se indica en la Figura 8, un componente fundamental de la presente propuesta, está en que las restricciones que se aplicaron sobre los conjuntos, se indicaron mediante el uso de expresiones algebraicas con el Language-Integrated Query (LINQ) del .Net Framework, generando recorridos matriciales (ciclos por filas y columnas) a los conjuntos.

El modelo de transporte implementado es un COP y también se utilizaron expresiones algebraicas para vincular la función objetivo (denominada AddGoal), indicándole que se deseaba una minimización. Aunque el modelo pudo ser resuelto por el sistema genérico de restricciones, se aprovechó que el modelo es lineal y se utilizó la directiva SimplexDirective, buscando reducir los tiempos de solución del modelo.

```
model.AddConstraint("Disponibilidad",
    Model.ForEach(fabricas,
    f => Model.Sum(Model.ForEach(distribuidores, d => x[f, d])) <= disponibilidad[f]));
model.AddConstraint("Demanda",
    Model.ForEach(distribuidores,
    d => Model.Sum(Model.ForEach(fabricas, f =>  x[f, d])) >= demanda[d]));

model.AddGoal("Meta",
    GoalKind.Minimize,
    Model.Sum(Model.ForEach(fabricas,
    f => Model.ForEach(distribuidores, d => costos[f, d] * x[f, d]))));

Solution solution = context.Solve(new SimplexDirective());
Report report = solution.GetReport();
```

Figura 8. Restricciones, función objetivo y directiva usada

VI. INTEGRACIÓN CON WINDOWS AZURE

El servicio de Web App dentro de Azure es una instancia de un web server configurada y optimizada para sitios web. Este servicio de Cloud Computing sirve de hosting, para la publicación de sitios web, propicios para realizar despliegues de servicios y aplicaciones, como se indica en la Figura 9.

El servicio de Web App debe ser contratado a través de Azure. La creación del servicio es automática y permite una configuración acorde a cada problema. En nuestro caso, la configuración por defecto logró satisfacer los requerimientos de procesamiento por parte de la implementación de código .

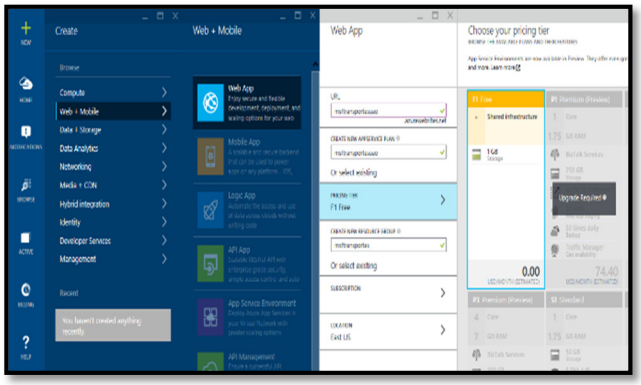

Figura 9. Adquisición instancia de Web App en Azure

La integración con Windows Azure fue muy sencilla de implementar, al ser Azure un servicio de Cloud Computing y que se consiguió una instancia del servicio llamado Web App.

La publicación se puede hacer de dos maneras: manual o a través de Visual Studio. En la solución propuesta se realizó la integración del servicio a través de Visual Studio. Para realizar esta integración fue necesario descargar el perfil de publicación (publish settings) directamente desde Azure; este archivo contiene los tokens que permiten la implementación y configura el cliente de conexión integrado con Visual Studio para la publicación del Web Service, como se aprecia en la Figura 10. Después de realizada la configuración, solo fue necesario publicar a través de la utilidad de Visual Studio, la cual se conecta de forma automática con el servidor, como se indica en la Figura 11.

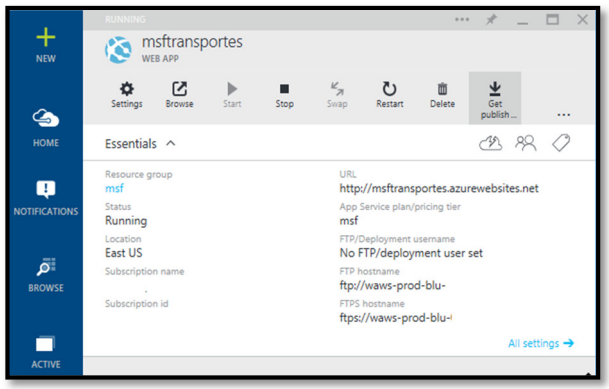

Figura 10. Descarga del archivo Publish Settings

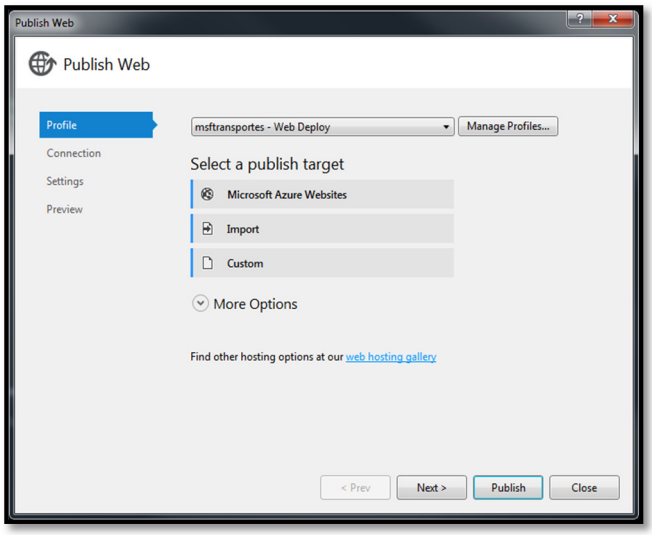

Figura 11. Configuración de publicación en Visual Studio

Como se indica en la Figura 12, para dejar disponible el Web Service, fue necesario configurar que el tipo del enlace (binding) sea wsHttpBinding y asegurarnos que el proyecto creara una copia local de la referencia de Microsoft Solver Foundation.

```
<system.serviceModel>
  <behaviors>
    <serviceBehaviors>
      <behavior>
        <serviceMetadata httpGetEnabled="true" httpsGetEnabled="true"/>
        <serviceDebug includeExceptionDetailInFaults="false"/>
      </behavior>
    </serviceBehaviors>
  </behaviors>
  <protocolMapping>
    <!--
      Ajuste del binding para que pueda ser usado en windows azure con framework 4
    -->
    <add binding="wsHttpBinding" scheme="http"/>
  </protocolMapping>
  <serviceHostingEnvironment aspNetCompatibilityEnabled="false" multipleSiteBindingsEnabled="true"/>
</system.serviceModel>
```

Figura 12. Archivo de configuración Web.config

En la Figura 13, se muestra que para realizar una prueba a la aplicación, fue necesario crear una aplicación cliente para consumir el Web Service alojado en Windows Azure. La aplicación pudo ser desarrollada bajo cualquier lenguaje de programación, con la única condición de que pudiera interpretar SOAP (Simple Object Access Protocol). En nuestro caso usamos una aplicación de consola con Visual Studio. A esta aplicación le adicionamos una referencia web, con la Url de la dirección del servicio web, la cual crea una clase cliente para consumir el Web Service . De igual manera fue necesario crear una instancia de la clase WCFTransporteClient que será a quien se le enviaran los parámetros de entrada al servicio y de la cual se obtendrá una respuesta.

```
StreamReader lector = new StreamReader(@"Ruta de archivo");
string xml = lector.ReadToEnd();
WCFTransporteClient cliente=new WCFTransporteClient();
Console.WriteLine(cliente.modeloTransporte(xml));
Console.ReadLine();
```

Figura 13. Aplicación cliente que consume el servicio en Azure

VII. RESULTADOS OBTENIDOS

Se logró modelar completamente un COP utilizando solo tecnología .Net, a través de Microsoft Solver Foundation, usando como lenguaje de programación C#, el cual tiene un buen desempeño en la obtención de la solución óptima, porque será resuelto utilizando el sistema de propósito específico para el computo de sistemas lineales. Aunque el modelo implementado contó con pocas restricciones, la implementación realizada puede operar con un modelo con mayor cantidad de restricciones con solo cambiar el número de datos suministrado en el archivo Xml suministrado como entrada, sin tener que modificar la codificación realizada.

La implementación realizada fue publicada como un Web service en Windows Azure, fácilmente desde Visual Studio, sin necesidad de configuraciones o procesos complejos.

VIII. CONCLUSIONES

Fue posible modelar un sistema de optimización de restricciones, usando el modelado, solución y posterior publicación como servicio en la nube, utilizando solo herramientas Microsoft, logrando un gran acople y disminución de tiempos de desarrollo e integración.
La implementación realizada es adaptable a modelos de mayor envergadura, permitiendo que sin modificar el código, pueda dar soluciones a problemas de transporte similares pero con mayor cantidad de variables de entrada y restricciones entre ellas.

MSF es una tecnología que debe ser tenida en cuenta para la solución de problemas reales de satisfacción u optimización de restricciones, ya que provee variedad de dominios y directivas, permitiendo en un mismo sistema la convivencia de solvers o modelos genéricos y específicos, permitiendo que con la modificación de una sola línea se obtengan soluciones de menor tiempo de ejecución en dichos casos específicos, pero permitiendo el modelamiento de sistemas genéricos de restricciones.

IX. TRABAJO FUTURO

Investigar en que tecnología Microsoft o de terceros se adicionaran las funcionalidades de MSF, ya hasta la versión 3.1 tuvo soporte por Microsoft.

REFERENCIAS

[1] R. A. Krzysztof, “Principles of Constraints Programming,” Eds. Cambridge University Press, 2009, pp. XX–XX.

[2] V. R. Peter, H. Seif, “Concepts. techniques, and models of computer programming,” Eds. MIT Press, 2004, pp. 749–776.

[3] Microsoft Solver Foundation, “What is Solver Foundation,” no publicado, incluido en la documentacion de MSF 3.0.2, pp.3–6.

[4] Microsoft Developer Network, “Microsoft Solver Foundation 3.1,” disponible en https://msdn.microsoft.com/en-us/library/ff524509(v=vs.93).aspx

[5] Microsoft Solver Foundation, "Microsoft Solver Foundation, Solver Foundation Services, Programming Primer", no publicado, Incluido en la documentación de MSF 3.0.2, pp.

[6] M. Tulloch, "Introducing Windows Azure" Eds. Microsoft Press, 2013, ISBN 978-0-7356-8288-7.

[7] I. S. Francisco, “Investigacion de operaciones,” vol. I, segunda edicion, ISBN: 958-8028-21-3, Eds. Corporacion Universitaria de Ibague, 2004, pp. 166–167.