MODELO DE APRENDIZAJE DE MAQUINAS PARA REDUCIR LAS FALLAS DE INSTANCIACIÓN EN COMPOSICIONES DE SERVICIOS

Byron Enrique Portilla Rosero [1], Jaime Alberto Guzmán Luna [2]

[1] Byron Enrique Portilla Rosero. Ingeniero de Sistemas, estudiante de la Maestría en Ingeniería de Sistemas en la Universidad Nacional de Colombia sede Medellín. beportillar@unal.edu.co
[2] Jaime Alberto Guzmán Luna**.** Magister en Ingeniería de Sistemas, profesor asistente Escuela de Sistemas Universidad Nacional de Colombia sede Medellín. jaguzman@unal.edu.co

Escuela de Sistemas, Universidad Nacional de Colombia – Sede Medellín
SINTELWEB: Grupo de Investigación "Sistemas Inteligentes Web"
Calle 59A No 63 – 20, Medellín, Colombia

RESUMEN

Este artículo, presenta un modelo de aprendizaje de maquinas orientado a reducir las fallas que se generan al asumir instancias de datos incorrectas por parte del planificador al momento de realizar el proceso de composición de servicios Web. Para ello, se enfatiza en la adquisición de información del mundo real a través de la ejecución del servicio y por medio de árboles de decisión, predecir la mejor regla de aprendizaje en función a la ejecución de un servicio.

Palabras clave: Composición de servicios Web, planificación, árboles de decisión, manejo de fallas.

Recibido: 5 de Octubre de 2009. Aceptado: 19 de Noviembre de 2009
Received: October 5, 2009 Accepted: November 19, 2009

A MACHINE LEARNING MODEL TO REDUCE INSTANTIATION FAILURES IN SERVICE COMPOSITION

ABSTRACT

This paper, show a machine learning model to reduce failures generated by assuming incorrect information instances by the planner during Web service composition process. For it, we emphasize in the acquisition of real-world information through the Web service execution and through decision trees, predicting the best learning rule according to the execution each service.

Keywords: *Web service composition; planning; decision trees; management failures.*

1. INTRODUCCIÓN

Son varias las investigaciones que se han enfocado en el uso de la planificación en inteligencia artificial como técnica práctica en la composición de servicios Web. [1], [2]. Sus distintas formas de implementación han contribuido en busca de mejores composiciones de servicios. Sin embargo, se ha encontrado un problema en aquellas implementaciones que utilizan una expansión de búsqueda hacia adelante, debido a que éstas requieren considerar todos los servicios, generando un conjunto de instancias irrelevantes por cada búsqueda para alcanzar un objetivo determinado. De manera que el espacio de búsqueda se incremente aún cuando se utiliza en la solución de pequeños problemas. [3], [4]. Esto conlleva a problemas en cuanto a malas selecciones de servicios para ser evaluadas al momento de ser ejecutados y también, se ve afectado el tiempo que tarda un planificador en instanciar todos los objetos y a su vez verificarlo en el mundo real.

Como solución a este problema, este documento propone un modelo de aprendizaje basado en árboles de decisión, el cual a partir de la información alcanzada en ejecuciones previas, trata de inducir al planificador al momento de la instanciación, mediante el uso de servicios que mejor sean representados en la ejecución. Es decir, seleccionar aquellos servicios donde sus instancias correspondan total o parcialmente con los valores de reales de ejecución, y de esta forma, controlar su acceso al grafo de planificación a través de un costo que representa el nivel en el cual, el servicio instanciado tendrá mejor aceptación por el grafo que otros servicios, evitando de esta forma la instanciación de servicios incorrectos para un determinado nivel del grafo.

Este documento está organizado de la siguiente forma: la sección 2, describe el algoritmo de planificación utilizado para la composición de servicios, la sección 3, presenta el modelo de aprendizaje de máquinas propuesto, la sección 4, muestra la integración del modelo de aprendizaje con el algoritmo de planificación, la sección 5, compara la propuesta presentada en este documento con investigaciones encontradas en el estado del arte y finalmente, la sección 6, presenta las conclusiones y trabajos futuros.

2. ALGORITMO DE PLANIFICACIÓN

En trabajos anteriores [5], [6], se presentó un agente de planificación de servicios Web en entornos dinámicos el cual, utiliza un algoritmo de planificación constituido por cuatro fases [7]: la primera fase consiste en el pre-proceso. En esta fase, se adquiere la información del dominio y problema de planificación a partir de la lectura de servicios semánticos descritos en la ontología OWL-S [8]; una vez almacenada esta información, se realiza la instanciación de operadores y predicados del dominio y de las especificaciones del problema. En este punto, se relaciona la instanciación a priori de todos los servicios con todo el conjunto de sus posibles valores. La segunda fase, consiste en la generación del grafo relajado de planificación (RPG) [9], el cual, describe una heurística para la construcción de planes. El RPG, es un grafo que consta de dos tipos de nodo, los literales y las acciones, y dos tipos de arista, las precondiciones y efectos. Este grafo inicia con la lectura del estado inicial, es decir, todos los literales ciertos para ese estado. Los niveles de acciones cubren todas las acciones aplicables al nivel anterior de literales y el nivel de literales, se extienden con la inclusión de los efectos positivos de dichas acciones. La expansión del RPG finaliza, al momento que se alcanza el nivel de literales que contiene todos los objetivos del problema, o cuando no es posible aplicar una nueva acción. Al instante que el planificador inicia esta fase, un conjunto de servicios instanciados es analizado de manera que puedan alcanzar el objetivo.

Como se menciona en la fase previa, el planificador toma como base el conjunto de servicios instanciados, para construir el grafo de búsqueda; esto implica en primer lugar, el aumento de tiempo en el procesamiento de información, cuando se trata un gran volumen de objetos bajo los cuales se instancia los servicios; y en segundo lugar, el aumento de fallas en la generación de grafo de planificación, debido a la selección de servicios instanciados con datos incorrectos para un nivel del grafo.

La tercera fase es la generación de planes mono-objetivo; ésta consiste en alcanzar cada uno de los objetivos de forma individual a través de planes independientes. El algoritmo, calcula de forma concurrente un plan para cada subobjetivos del problema y finalmente se lleva a acabo. Aunque estos planes son incompletos, estos se van

refinando hasta alcanzar un plan válido. Finalmente, está la fase de ordenación de planes; ésta, permite identificar cual es el servicio que debe ser ejecutado entre todos aquellos que conforman cada uno de los planes.

Aunque todas las fases descritas anteriormente son determinantes en la planificación, este documento se centra en las fases 1 y 2 del algoritmo. Es aquí donde se interviene el algoritmo de planificación. Por lo tanto, se plantea la solución de utilizar mecanismos de aprendizaje de máquinas que actúe sobre estas fases y por medio de la experiencia adquirida a través de la ejecución, se optimice el algoritmo de planificación para alcanzar mejores resultados en futuras composiciones de servicios Web.

3. MODELO DE APRENDIZAJE

La instanciación de servicios dentro del proceso de planificación se encuentra asociada a un conjunto de niveles dentro del grafo de planificación, que establece la posición que tomará un servicio dentro de un plan; de esta forma se puede establecer cuáles son los servicios que serán incluidos o excluidos según su costo.

Para determinar el costo que tendrá cada servicio, el modelo emplea un monitor en la ejecución del servicio que verifica el paso de mensajes en los cuales, se analiza cada una de las instancias que un servicio requiere para ser ejecutado en un nivel particular del grafo. El seguimiento de las instancias, conduce a identificar si el servicio se pudo ejecutar o no.

Por lo tanto, el modelo de aprendizaje, está asociado directamente a la ejecución de servicios. La información que ésta retorna, determina la validez del mismo. Una ejecución *e* asocia la tupla *t(S,I,O)* donde *S* representa el servicio que será ejecutado. *I*, son los datos de entrada que ejecutan el servicio y *O*, representa los valores de salida con relación a las entradas del servicio. La verificación que realiza el ejecutor, es definir si las entradas y salidas asociadas a un servicio por parte del planificador, son la representación real del mundo en ejecución, clasificándolo entre ejecutable y no ejecutable. De esta forma, la ejecución juega con el conjunto de servicios instanciados hasta encontrar una solución posible pero el tiempo asociado al proceso de planificación y ejecución, puede crecer exponencialmente dependiendo del número de instancias posibles para un servicio. Por lo tanto, la interpretación de las entradas y salidas se convierte en la regla de aprendizaje para identificar cuál es el conjunto de datos de salida que conllevan a una ejecución satisfactoria utilizando el menor tiempo posible.

El modelo de aprendizaje genera una regla *R* que toma la información del mensaje (SOAP) de planificación y la asocia a los valores de respuesta de la ejecución. *R(S,I,O,c)* y adiciona un dato que clasifica la validez del servicio. Es decir, qué tan exitoso o fallido fue el servicio ejecutado según los valores de clasificación (bueno, posible, malo).

Bueno, cuando las salidas del planificador coinciden con las salidas del servicio después de ejecutarlo.
Malo, cuando ninguna de las salidas esperadas por el planificador, coinciden con las salidas del servicio después de ejecutarlo.
Posible, cuando al menos una de las salidas del planificador, coincide con alguna salida del servicio después de ejecutarlo.

Cada vez que el planificador envía un servicio para ser ejecutado, se captura la tupla *t* de ejecución en la cual, se representa la clasificación para dicho servicio. Con esta información, se establece un conjunto de valores disponibles para el aprendizaje y sobre los cuales, se realiza el proceso de predicción. Para ello, se utiliza el algoritmo de árboles de decisión [10]. Este, describe el conjunto de relaciones que hay entre las entradas y posibles salidas de un servicio a partir de particiones realizadas recursivamente. Para apoyar la decisión de este algoritmo se aplica la herramienta TILDE [11] que implementa una extensión del los árboles de decisión C4.5 [12] en la cual, se predice el valor de un determinado atributo en relación a una información.

Para presentar el árbol de aprendizaje es necesario contar con las entradas:

Lenguaje bias, identifica el objetivo de predicción y el conjunto de instancias conocidas. Estas dos definiciones son adquiridas a través del dominio y problema de planificación. En la Fig. 1, se presenta el lenguaje de inducción para el servicio getiteminformation del dominio shopping, el cual trata de conseguir la información del item 12303. Así: la parte superior, representa que el objetivo a

predecir, es la ejecución del servicio getItemInformation, este se compone de tres parámetros, el primero *example*, representa un identificador único que se le asigna después de ser ejecutado el servicio; el segundo parámetro *input*, determina las entradas requeridas por el servicio para ser ejecutado. Finalmente el último parámetro es la clase de ejecución *class*. Es decir, determina el tipo de ejecución del servicio (bueno, posible, malo). Por otra parte, se describe las instancias conocidas en el problema de planificación, para este caso, estas corresponden a las instancias asociadas a las salidas del servicio getItemInformation como son mp4, iron entre otras y las cuales, cuentan con un identificador único que representa la ejecución del servicio.

```
% Objetivo a predecir
type(execution_getItemInformation(example,input,class)).
class([bueno,malo,posible]).

%Salidas Instanciadas

type(microwave(example)).
type(mp4(example)).
type(iron(example)).
type(plasma_tv(example)).
type(blender(example)).
type(mp4_acme3(example)).
type(microwave_acme4(example)).
type(iron_acme1(example)).
```

Fig. 1. Lenguaje bias para el dominio de compras shopping

La base de conocimiento. Contempla el conjunto de las tuplas *t* de ejecución (instancias de entrenamiento). Este conjunto de datos es extraído en el proceso de ejecución del servicio. La Fig. 2. muestra las tres posibles clasificaciones para el servicio getiteminformation que tiene como entrada el item 12303. Donde cada ejemplo, describe la versión instanciada del lenguaje bias Fig 1. Es decir, el servicio ejecutado, su identificador de ejecución, su entrada(s) y su clasificación, además las salidas producidas en la ejecución del mismo.

```
%Ejemplo 1
  execution_getItemInformation(e1,Item12303,bueno).
  iron(e1).
  iron_acme1(e1).
%Ejemplo 2
  execution_getItemInformation(e2,Item12303,malo).
  blender(e2).
  [illegible](e2).
%Ejemplo 3
  execution_getItemInformation(e3,Item12303,posible).
  iron(e3).
  mp4_acme3(e3).
```

Fig. 2. Base de conocimiento después de la ejecución

A partir de la información anteriormente descrita, se genera el aprendizaje del árbol de decisión donde cada rama presenta una regla de ejecución de cada servicio. Los nodos internos contienen el conjunto de condiciones bajo las cuales, la regla es verdadera; cada hoja representa la clasificación de acuerdo a los tres parámetros mencionados anteriormente. Además, se obtiene el número de ejemplos que son cubiertos por esa regla para esa clasificación como se presenta en la Fig. 3.

```
execution_getiteminformation(-A,-B,-C)
iron(A) ?
+--yes: [posible] 34.0 [[bueno:0.0,posible:14.0,malo:20.0]]
+--no:  iron_acme1(A) ?
        +--yes: [posible] 6.0 [[bueno:0.0,posible:6.0,malo:0.0]]
        +--no:  [malo] 20.0 [[bueno:0.0,posible:0.0,malo:20.0]]
```

Fig. 3. árbol de decisión para el servicio getiteminformation del dominio de compras shopping.

La Fig. 3, presenta el árbol de aprendizaje en función de las entradas del servicio getiteminformation. De acuerdo a este árbol, cuando se tiene como entrada el item12303 para este servicio, existe la posibilidad que ese item sea iron o que sea iron de marca acme1.

4. INTEGRACIÓN APRENDIZAJE Y PLANIFICACIÓN

Para integrar los datos mencionados en el árbol de aprendizaje Fig. 3., con el algoritmo de planificación descrito en la sección 2, en primer lugar se evalúa cada una de las reglas generadas en el árbol, estas reglas son generadas en lenguaje prolog. Para el caso de la figura 3, las reglas son:

Regla 1. getItemInformation(A,[posible]) :- iron(A), !.

La regla 1, muestra que hay una posibilidad que el servicio se ejecute correctamente cuando una de sus salidas es iron.

Regla 2. getItemInformation(A,[posible]) :- iron_acme1(A), !

La regla 2, muestra que hay una posibilidad que el servicio se ejecute correctamente cuando una de sus salidas es iron_acme1.

Asimismo, se calcula el número de ejemplos que cubre cada regla. Es decir, la primera regla, cubren 14 ejemplos de 34 posibles y para la segunda regla, cubre 6 de 6 ejemplos posibles Fig. 3.

Una vez se conozcan estos valores, la inserción de estos al problema de planificación, se convierte en un problema de costos, definiendo como métrica la relevancia de la información.

$$revInfo - \frac{esc}{tec} \quad (1)$$

Donde, *esc* es el conjunto de ejemplos exitosos cubiertos por la regla y *tec* es el conjunto total de ejemplos ejecutados. Estos valores entran a formar parte de la generación del grafo de servicios instanciados, de manera que se cubran principalmente aquellos servicios que tengan una regla buena o posible que cubra sus salidas. Así, se representaran los servicios que se aproximan a la realidad de ejecución.

Para el servicio getiteminformation según la Fig. 3. las reglas de aprendizaje tendrán los siguientes valores: cuando el servicio tenga como entrada el item12303 y tenga una salida iron el valor de revInfo será de 0.41% mientras que si tiene como salida iron_acme1 el valor de la métrica revInfo será 0.1%.

Las fases 1 y 2 del algoritmo de planificación se verán afectadas con la inserción de un nuevo valor métrico que reorganice el modelo de instanciación de los servicios. En la Fig. 4. Se observa el código de programación utilizado para intervenir el algoritmo de planificación el cual, valida todas las reglas del árbol de aprendizaje y asigna su valor a la métrica relevancia de información de cada servicio asociado a esa regla.

```
//Asignación de la métrica de aprendizaje revInfo para cada servicio
...
    revInfo(s_i)   r(s_i)
...
```

Fig. 4. Asignación de costos para un servicio

En la fase del pre-proceso, la métrica de aprendizaje *revinfo* es integrada a cada servicio sobre el cual, se tiene una regla de inducción que lo cubra. Así, se identifica los servicios que deben ser seleccionados por el planificador en la generación del grafo e implícitamente inducir a que estos servicios sean los primeros en ejecutarse. De esta manera, cuando el grafo de planificación realiza la ordenación de los niveles proposicionales, cada servicio queda asociado a un nivel particular en función de su costo siendo más representativo para el grafo, aquel servicio con menor costo.

$$cost(s) = evalue(m, result(s, S0, 0)) - evalue(m, S0) + revInfo(s) \quad (2)$$

El costo para el servicio *s* se calcula como el incremento en el valor de la métrica *m* causado por la ejecución del servicio *s* en el estado actual So, más el valor adquirido del aprendizaje.

5. TRABAJOS RELACIONADOS

En la literatura se ha encontrado varios trabajos orientados al uso de técnicas de aprendizaje que apoyen los modelos de planificación como en [13], el cual, se enmarca en función de proveer a planificadores, la información de la ejecución de las acciones para alcanzar planes robustos. Es decir, la utilización de una métrica que determine la funcionalidad de un plan, haciendo referencia a la identificación de las acciones de éxito. Para ello, define tres fases: la primera consiste en captura de la información; esto es obtener a través de una serie de ejecuciones, el conjunto de información que representa el comportamiento de una acción en el mundo real. La segunda fase, consiste en la inducción de la información encontrada, más específicamente los patrones de inducción (aquellas características que determina el éxito de la ejecución) e implementa la programación lógica inductiva, mecanismo de aprendizaje de máquinas que le permite generar patrones de aprendizaje. Finalmente, utiliza dichos patrones en la generación del plan con robustez. En este caso se definen dos posibilidades para calcular la robustez de un plan. La primera es la utilización de efectos condicionales probabilísticos para determinar la robustez de un plan, y la segunda es a través del uso de los costos de planificación a través de condiciones de costos. Esta última se determina a través de la fragilidad (verificar la robustez de un plan), la cual, permite transformar la maximización del producto de las probabilidades de éxito de acción a lo largo del plan, en una minimización de la suma de los costos de fragilidad. Se calcula como –log(p) donde p es la probabilidad. Este trabajo se orienta en la adquisición de mejores planes, en función de acciones de planificación que cambian el mundo.

Asimismo en [14], se presenta un modelo de aprendizaje basado en el algoritmo boosting. Este modelo, permite aprender reglas de selección de acciones ponderadas. Es decir, un peso numérico es asignado a una acción en un estado de transición el cual, es usado para guiar las estrategias de búsqueda hacia delante de un planificador, convirtiéndose en un problema de calificación de acciones y de esta forma alcanzar un objetivo dentro de un problema de planificación de manera robusta. En [15], se muestra un enfoque en la adquisición de macro acciones, las cuales, dependen de métodos de aprendizaje de maquinas para sugerir macros que contemplan las instancias de un problema de planificación. En este trabajo se pretende mejorar el promedio de ejecuciones. El aprendizaje de macros con instancias especificas, se orienta a la estimación de una subserie meta-acciones que mejor representen la instanciación de un problema de manera que un predictor pueda decidir cuáles son los macros que deben ser integrados al dominio de planificación.

Por otro lado, están los modelos de aprendizaje orientados a la composición de servicios. En el trabajo de [16], se presenta un algoritmo de aprendizaje basado en la búsqueda primero en profundidad (LDFS), para alcanzar óptimas composiciones. Para esto, proponen el uso de información adicional de los servicios teniendo en cuenta sus requerimientos funcionales y no funcionales; estos últimos, representan las métricas de calidad para cada servicio (QoS) como el tiempo y el volumen de información soportada por cada uno de estos. Definen un servicio con la tupla (I,O,Q) donde I, son las entradas, O las salidas y Q, el conjunto de métricas de calidad evaluadas. De esta manera, identifican la secuencia que minimice el número de servicios por cada estado, donde se optimice la composición de los servicios en función de las calidades asociadas. De manera que se invoca un servicio que cumpla con los requisitos solicitados y que además, maximice su calidad. Por otro lado, el trabajo de [17], expone un aprendizaje de definiciones semánticas a través de procesos de inducción. Las definiciones son dadas en el sistema o aprendidas previamente; esto, mediante el uso de ejemplos entrenados. La característica que evalúa es el uso único de las entradas y salidas donde se identifican los tipos de datos que se requieren en la ejecución, y cuál es el tipo de dato que se espera. Para esto, utiliza la programación lógica inductiva con el objeto de adquirir las reglas de conocimiento.

A diferencia de las investigaciones mencionadas, este trabajo se orienta al descubrimiento de relaciones de datos a partir de servicios de información, tratando de conseguir mejores planes para la ejecución, en el menor tiempo posible, evaluando el conjunto de posibles fallas generadas en la instanciación de un grafo de planificación.

6. CONCLUSIONES

En este trabajo, se presentó un modelo de aprendizaje basado en árboles de decisión, el cual permite aprender un conjunto de instancias asociadas a la ejecución de servicios Web, con el objeto de identificar reglas de decisión que ayuden al planificador a utilizar aquellos servicios que mejor sean representados en la ejecución. Para ello, se utilizó la métrica <relevancia de información> la cual, interviene en la asignación del costo de un servicio durante la generación del grafo de planificación.

Actualmente se está desarrollando la versión inicial del planificador supervisado por el mecanismo de aprendizaje y, se está trabajando en la identificación y control de fallas al interior de la composición de servicios en cuanto al uso de servicios de cambio de mundo.

7. AGRADECIMIENTOS

El presente trabajo está apoyado parcialmente por el proyecto de investigación de tesis de maestría de la Escuela de Sistemas de la Universidad Nacional de Medellín “Modelo Basado en Aprendizaje de Máquinas para el Manejo de Riesgo de Falla Durante la Composición de Servicios Web” noviembre 2008.

8. REFERENCIAS BIBLIOGRÁFICAS

[1] Klusch M. Semantic service discovery and composition: A survey. En: M. Schumacher, H. Helin, H. Schuldt (Eds.): CASCOM - Intelligent Service Coordination in the Semantic Web. Chapter 4. Birkhaeuser Verlag, Springer, 69-114, 2008.

[2] Marconi A., Pistore M., Poccianti P., Traverso P. Automated Web service composition at

Work: the Amazon/MPS Case Study, The 11th IEEE International Conference on Web Services (ICWS), Utah, USA, July 2007.
[3] Ghallab M., Nau D., Traverso P. Automated Planning Theory and Practice, Morgan Kaufmann, 2004.
[4] Agarwal V., Chafle G., Mittal S. Understanding approaches for Web service composition and execution. En: Proceedings of the 1st Bangalore Annual Compute Conference, Compute 2008, Bangalore, India, January 2008.
[5] Guzman J. and Ovalle D. Web services planning agent in dynamic environments with incomplete information and time restrictions, The 11th IEEE International Conference on Computational Science and Engineering – Workshops, Sao Paulo, Brasil, July 2008.
[6] Guzman J. and Ovalle D. Reactive planning model for Web service composition under time restrictions, En Third International Conference on the Digital Society (ICDS 2009), Cancun, Mexico, February 2009.
[7] Guzman J. y Ovalle D. Un modelo de planificación incremental para servicios Web semánticos, Revista Avances en Sistemas e Informática, Vol.4 No. 3, Medellín, Diciembre 2007.
[8] OWL-S Coalition: OWL Web Services 1.1, Disponible en: http://www.daml.org/services/owl-s [consultado el 20 de agosto de 2009]
[9] Hoffman, J., Nebel, B.: The FF planning system: fast planning generation through heuristic search. En: JAIR. 14, reading 253-302, 2001.
[10] Quinlan J. R. Induction of Decision Trees. In, Machine Learning Vol.1 reading 81-106, 1986.
[11] Blockeel H. and Raedt L. D. Top-down induction of firstorder logical decision trees. Artificial Intelligence 101(1-2):285–297. June 1998.
[12] Salzberg S. L. Book Review: C4.5: Programs for Machine Learning by J. Ross Quinlan. Morgan Kaufmann Publishers, Inc., 1993, Machine Learning, Vol.16 No. 3, reading .235-240, 1994.
[13] Jimenez S. Learning actions success patterns from execution. In Doctoral Consotium of the International Conference on Automated Planning & Scheduling, ICAPS'07. Rhode Island, USA, 2007.
[14] Xu Y., Fern A. and Yoon S. Learning Weighted Rule Sets for Forward Search Planning. En: Workshop on Planning and Learning, ICAPS-2009. Thessaloniki, Greece, September 2009.
[15] Alhossaini M. and Beck J. Learning Instance-Specific Macros. En: Workshop on Planning and Learning, ICAPS-2009. Thessaloniki, Greece, September 2009.
[16] Nam W., Kil H. and Lee J., QoS-DrivenWeb Service Composition Using Learning-based Depth First Search, IEEE Conference on Commerce and Enterprise Computing, Vienna, Austria, July 2009.
[17] Carman M. and Knoblock C. Learning Semantic Descriptions of Web Information Sources Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07). Hyderabad, India, January 2007.