ISSN 0717-9103
ISSN Online 0718-8307
Ingeniería Industrial - Año 8 N° 2: 63 - 74, 2009

PROGRAMACIÓN CON RESTRICCIONES DINÁMICA*

DYNAMIC CONSTRAINT PROGRAMMING

Broderick Crawford[1,2],Carlos Castro[2],Eric Monfroy[2,3]
[1]Pontificia Universidad Católica de Valparaíso, Valparaíso. Chile.
[2]Universidad Técnica Federico Santa María, Valparaíso. Chile
[3]LINA, Université de Nantes, France.

RESUMEN

Este trabajo presenta algoritmos de resolución de Problemas de Satisfacción de Restricciones, capaces de medir el desempeño de su proceso a través de indicadores relevantes, posibilitando su auto-ajuste. Las posibilidades de adaptación tienen relación con cambiar la Estrategia de Enumeración en uso al detectarse un mal rendimiento. El propósito es encontrar soluciones rápidamente para diferentes tipos de problemas, solucionando una de las limitantes en torno a las Estrategias de Enumeración: “para un problema dado, se tiene una estrategia particular que funciona bien, pero limitada en la resolución eficiente de otros problemas”.
La propuesta descrita se inspira en enfoques adaptativos existentes, pero que han sido diseñados con otra orientación, como el caso de la Satisfacción de Restricciones Adaptativas, donde dada una secuencia de algoritmos a utilizar, los malos son detectados y reemplazados por el próximo candidato. Sin embargo, en este trabajo no se pretende cambiar un algoritmo completo, sino para un mismo algoritmo de resolución modificar la estrategia que lo guía en base a la observación del conjunto de indicadores, obtenidos del análisis del mismo proceso de búsqueda que está desempeñando.

Palabras Claves: Programación con restricciones, optimización combinatoria, problemas de satisfacción de restricciones.

ABSTRACT

This paper presents algorithms for solving Constraint Satisfaction Problems (CSPs) capable of measure the performance of their process through relevant indicators to enable adaptations (self-adjust). The adaptation options are related to the change of Enumeration Strategy, used when bad performance is detected. Done in this way, to find quickly solutions to different problems, solving one of the limitations on the Enumeration Strategy: “for a given problem, there is a particular strategy that works very well, but limited in the efficient resolution of other problems.”
The proposal outlined is based on existing approaches, that have been designed with other orientation, such as Adaptive Constraint Satisfaction (ACS), where given a sequence of algorithms to use, bad ones are detected and replaced by the next candidate. However, this paper is not intended to change a complete algorithm, but for the same algorithm, modify the strategy that guides it based on the observation of the set of indicators obtained from the analysis of the search process that is playing.

Keywords: Constraint programming, combinatorial optimization, constraint satisfaction problems.

*Este trabajo fue presentado en el VIII Congreso Chileno de Investigación Operativa, realizado por la Universidad del Bio-Bio en la ciudad de Chillán en Octubre del año 2009

Autor para correspondencia: broderick. crawford@ucv.cl

Recibido: 27.04.2009 Aceptado: 18.11.2009

INTRODUCCIÓN

La Programación con Restricciones, en inglés Constraint Programming (CP), ha sido definida como una tecnología de Software utilizada para describir y solucionar de manera efectiva grandes y complejos problemas, particularmente combinatoriales (Barber & Salido, 2003; Barták, 1998). Dichos problemas pueden modelarse como un Problema de Satisfacción de Restricciones (Apt, 2003), en inglés Constraint Satisfaction Problem (CSP), el cual consiste en un conjunto de variables, cada una con un dominio asociado, más un conjunto de restricciones. Los dominios son un conjunto de posibles valores para las variables. Cada restricción es definida sobre un conjunto de variables, y restringe de alguna manera la combinación de valores que pueden ser asignados a esas variables. Resolver un CSP consiste en encontrar una asignación de valores a las variables, de manera tal que no se viole ninguna restricción (Zoeteweij, 2005). Actualmente, la comunidad de Programación con Restricciones resuelve estos problemas utilizando una aproximación completa que consiste en alternar fases de propagación y enumeración (Barták, 1998; Castro *et al.*, 2005b). El proceso consiste en la exploración sistemática de todas las posibles asignaciones de valores a las variables, formando un árbol de búsqueda. La propagación de restricciones poda el árbol de búsqueda eliminando valores que no contribuyen a alguna solución. La enumeración crea una rama instanciando una variable y otra rama cuando la primera no se ha podido satisfacer. Entonces, al ejecutar una fase de enumeración deben resolverse dos situaciones: seleccionar la variable a ser enumerada y seleccionar el valor a asignar a dicha variable. Para la primera decisión se utiliza una Heurística de Selección de Variable, encargada de determinar el orden en que se enumerarán las variables, y para la segunda elección se utiliza una Heurística de Selección de Valor.

En conjunto, una Heurística de Selección de Variable y una Heurística de Selección de Valor, constituyen lo que se conoce como una Estrategia de Enumeración (Van Roy & Haridi, 2004), que se encarga de guiar la fase de enumeración. En el ámbito de la Programación con Restricciones, estas estrategias son cruciales en el rendimiento del proceso de resolución, donde una elección adecuada puede reducir considerablemente el costo computacional de buscar una solución a un CSP.

Existen numerosos estudios referentes a estrategias de enumeración (Beasley, 1990; Castro *et al.*, 2005a; Prosser *et al.*, 2003), algunos centrados en definir criterios generales (ej. first fail o dominio mínimo) para selección de variables, y para selección de valor (ej. valor mínimo, valor máximo o aleatorio); otros en cambio se han enfocado en especificar estrategias para una clase dada de problemas (Norman *et al.*, 1996; Caseau & Laburthe, 1994); respecto a esto último, existen trabajos destinados a determinar la mejor estrategia basada en algún criterio estático (Beck *et al.*, 2004; Sturdy, 2003); es decir, el criterio de selección es determinado una única vez antes del inicio del proceso de resolución, y permanece inalterable durante todo el proceso. Sin embargo, es sabido que tomar una decisión a priori, referente a una buena selección de variable y de valor, es trabajo difícil, ya que los efectos de la estrategia pueden ser imprevisibles. En este último tiempo se han hecho esfuerzos en determinar buenas estrategias en base a información generada durante el proceso de resolución. Sin embargo, el establecimiento de qué información medir y cómo guiar la búsqueda, es un campo de investigación abierto.

Atendiendo a lo expuesto anteriormente, en este trabajo se plantea incorporar a los algoritmos de búsqueda (principalmente del tipo Look-ahead) técnicas que permitan la identificación y medición de indicadores relevantes del proceso de resolución, los cuales hagan posible la clasificación del estado del proceso de ejecución, entendido como el progreso de la resolución, y así lograr determinar si la actual estrategia utilizada tiene un mal rendimiento y si es preciso cambiarla por otra que pueda mejorar dicho rendimiento. Con esto no se busca mejorar la

resolución de un solo problema, sino que se está interesado en encontrar eficientemente soluciones para un espectro amplio de diferentes tipos de problemas, a través de la observación de indicadores que permitan reaccionar frente a la elección de una mala estrategia. Los algoritmos a desarrollar podrán readecuar (auto-ajustar) su proceso de búsqueda, aun si la estrategia inicial no es la más adecuada para el problema que se está resolviendo.

El marco general de las técnicas de resolución empleada es la Programación con Restricciones, que consiste fundamentalmente en alternar fases de propagación y enumeración, según una estructura algorítmica similar a la mostrada en la Figura 1. La *Propagación de restricciones* reduce el dominio de las variables, removiendo aquellos valores que no pueden participar en la solución; rigurosamente hablando, la Propagación de restricciones fuerza algún nivel de consistencia local, tal como arco consistencia o camino consistencia (Apt, 2003; Castro *et al.*, 2005b; Zoeteweij, 2005). Sus instancias más conocidas son Forward Checking, que asegura la consistencia de las variables relacionadas (que participan en una misma restricción) con la variable recientemente enumerada, y Look-Ahead, que fuerza la consistencia de un CSP completo (Apt, 2003; Barták, 1998).

Por su parte la *Enumeración* consiste en tomar un CSP y dividirlo en dos CSPs más pequeños, hasta alcanzar alguno de ellos el estado solucionado o fallido; de esta forma, si se tiene un CSP P, la etapa de enumeración sobre una variable crea un CSP P' donde un valor es asignado a esta variable y otro CSP P'' con el resto de valores del dominio de la variable. En esta fase es donde se utiliza la estrategia de enumeración, constituida por las heurísticas de selección de variable y valor, las cuales ayudan a determinar qué variable elegir en un determinado momento y qué valor asignarle a dicha variable. De esta forma, la función *Enumeración* de la Figura 1 es la aplicación de la estrategia seleccionada y *Proceder por casos* es la función que administra la elección de los puntos creados por la enumeración, lo cual permite la exploración del árbol de búsqueda. En general, la estrategia de enumeración se especifica al comienzo del proceso de resolución y permanece constante durante toda la ejecución de dicho proceso. En la literatura existen criterios generales de selección de variable y valor basados en principios clásicos como Naive, First-Fail, Promise y otros (Beck *et al.*, 2004), los cuales sirven de base a la investigación aquí planteada. Sin embargo, existe muy poco estudio referente a la utilización de otras técnicas (adaptativas, reactivas, procesos estocásticos y metaheurísticas) como criterios de selección de variable y/o selección de valor. Así lo demuestra el foco de atención puesto por investigadores en Programación con Restricciones en sus últimos proyectos (consultar resultados Fondecyt 2006 a 2008).

```
WHILE no solucionado OR fallido
    Propagación de restricciones
    IF no fallido AND no solucionado THEN
        Enumeración
        Proceder por casos
    END
END
```

Figura 1: Algoritmo de Resolución

Importancia de las Estrategias de Enumeración

La Programación con Restricciones, entendida como tecnología, ha alcanzado numerosos éxitos en lo que a aplicaciones industriales se refiere. Uno de los primeros usos desarrollado

en 1990 fue la planeación del puerto de containers en Hong Kong; la aplicación fue construida usando restricciones de dominio finito. Otro de los usuarios iniciales fue Siemens, que aplicó restricciones booleanas a los problemas de diseño y de integración de circuitos. Siemens y Xerox ahora están aplicando restricciones a problemas de control en tiempo real (Van Roy & Haridi, 2004).

Otras áreas donde la Programación de Restricciones ha tenido mucho éxito incluyen Scheduling, Rostering y problemas de transporte. Las restricciones son utilizadas para planificar la producción en industrias químicas, refinerías de petróleo, planificación en la industria de la aviación, la minería y el transporte. También se ha utilizado para planificación financiera y gestión inversiones.

Todos estos problemas industriales/reales involucran una gran cantidad de variables, valores que pueden ser asignados a esas variables y restricciones que hay que satisfacer de manera simultánea, para lo cual la Programación con Restricciones proporciona diversas técnicas que los permiten modelar y solucionar. Estas técnicas son susceptibles de perfeccionar/potenciar mediante la realización de investigaciones y propuestas como las aquí planteadas, de tal manera de alcanzar procesos de resolución cada vez más eficientes y usables.

Encontrar una solución a un CSP que dé satisfacción plena a todas las restricciones involucra necesariamente llevar adelante un proceso de búsqueda bastante costoso, que en la actualidad está siendo abordado con las técnicas de propagación de restricciones y enumeración ya descritas. En este proceso de búsqueda, específicamente en la fase de enumeración, la estrategia utilizada como guía tiene un efecto importante en el rendimiento del proceso de resolución. Para visualizar mejor la importancia de las estrategias de enumeración, en la FIGURA Nº2 se muestran los árboles de búsqueda de la resolución del problema 10-Reinas (Apt, 2003; Barták, 1998) utilizando tres distintas estrategias de enumeración.

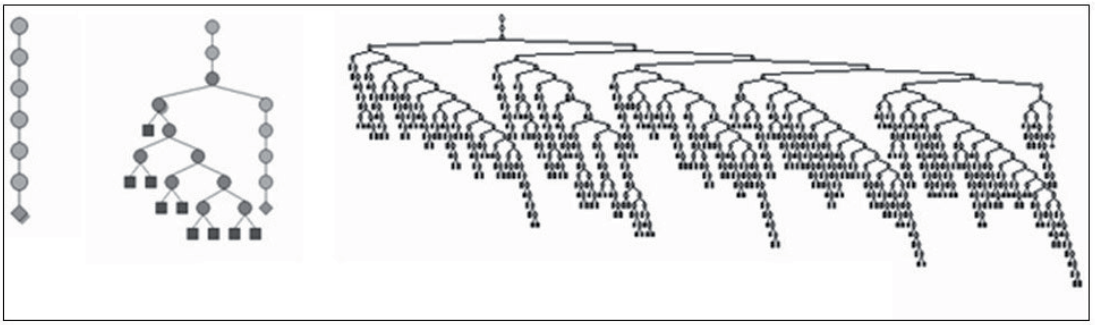

Figura Nº2: 10 Reinas Solucionado con 3 Estrategias Diferentes

En la FIGURA Nº2 es posible ver que la primera estrategia utilizada encuentra inmediatamente la solución. La segunda estrategia, después de algunas malas elecciones (de variable y valor), finalmente obtiene directamente una solución; en cambio, la tercera estrategia de enumeración realiza numerosas malas elecciones antes de obtener una solución.

Observando esto, es obvio que diferentes estrategias tienen rendimientos significativamente diferentes, por lo cual es crucial seleccionar una buena estrategia de enumeración. Sin embargo, ésta no se puede predecir de manera general, y es por esto que se considera relevante observar el proceso de resolución, identificar y clasificar distintos estados de dicho proceso en base a indicadores significativos, para con esta información poder reaccionar oportunamente frente a una mala decisión.

Estrategias de Enumeración Adaptativas

Se pretende con este trabajo solucionar la limitante que trae el hecho de que la estrategia de enumeración permanezca constante durante todo el proceso de resolución. Una de las limitaciones que esto acarrea es el hecho de que una estrategia de enumeración funciona eficientemente para un conjunto bastante acotado de problemas, y en el peor de los casos para un único problema (super especialización). Para esto, se propone que los algoritmos a desarrollar en base a las técnicas de enumeración y propagación ya mencionadas se adapten al problema mientras se ejecute su proceso de resolución. Adaptarse, en el sentido de lograr determinar si la estrategia de enumeración elegida al inicio del proceso está teniendo un buen desempeño; si no es así, el algoritmo debe ser capaz de detectar dicha situación y cambiar la estrategia.

Para materializar lo anterior, se realizan observaciones continuas (medición de indicadores previamente definidos) sobre el proceso de resolución, las cuales proporcionan información indicando el estado del proceso de resolución, este último entendido como el grado de avance o progreso en la resolución del problema. De manera simple, si dicho proceso resulta favorable, se continúa utilizando la misma estrategia de enumeración; de lo contrario, el algoritmo reacciona y cambia la estrategia por una "más prometedora". Las observaciones realizadas sobre el proceso de resolución y la información allí obtenida reciben distintos nombres en la literatura. Es así como en Borret *et al.* (1996) y Monfroy *et al.* (2006), estas observaciones reciben el nombre de monitors, snapshots, indicadores o simplemente mediciones. Cualquiera sea la denominación que se utilice, el objetivo de éstos es reflejar el progreso en la resolución de un problema, permitiendo elaborar un juicio respecto del funcionamiento del proceso de resolución, por lo cual estos indicadores deben ser simples y cuantitativos; pudiéndose utilizar uno o más dependiendo de la técnica utilizada y/o el problema a resolver. También se pueden usar combinaciones o porcentajes de unos respecto a otros. Algunos ejemplos de la información a medir son:

- Tiempo de ejecución
- Número de backtracks
- Número de nodos visitados
- Profundidad máxima alcanzada en el árbol de búsqueda
- Profundidad del nodo actual
- Número de variables instanciadas en un determinado momento
- Número de variables sin instanciar en un determinado momento
- Tamaño del espacio de búsqueda
- % variables instancias sobre el total de variables
- % de profundidad alcanzado
- % de valores del dominio ya "intentados" para una variable

En resumen, se pretende traducir el desempeño y/o comportamiento de las estrategias de enumeración en términos de indicadores y sus rangos de aceptabilidad, de modo de reaccionar frente a un mal rendimiento de dicha estrategia y cambiarla, con el objetivo final de mejorar la

resolución de un espectro amplio de diferentes tipos de problemas. Tomando las características y el concepto de indicadores dado, es posible hacer una analogía con Balanced Scorecard (Norton & Kaplan, 1996), metodología de trabajo que ayuda a las organizaciones a traducir la estrategia en términos de mediciones, de modo que impulse el comportamiento y el desempeño organizacional hacia el logro de los objetivos estratégicos. Este sistema de apoyo a la gestión, junto con incluir conceptos como indicadores o mediciones, objetivos estratégicos, entre otros, proporciona el concepto de medios, es decir, acciones o iniciativas que se requieren desarrollar para lograr las metas u objetivos. Dicho concepto resulta ser importante para la investigación aquí propuesta, y se relaciona con las acciones a seguir una vez decidido el cambio de estrategia. Estas acciones, según lo expuesto en Borret *et al.* (1996) y Monfroy *et al.* (2006), pueden ser:

- Acciones de Recomenzar (Restart): esta acción (la considerada en este trabajo) reinicia completamente la búsqueda una vez que los indicadores sugieren que la actual estrategia de enumeración no se ha comportado eficientemente.

- Acciones Constructivas (Constructive): en esta acción, la información generada hasta este momento sobre la búsqueda, se mantiene y sigue siendo utilizada después de realizar el cambio de estrategia de enumeración.

- Metabacktracks: esta acción deshace parte de la búsqueda hasta niveles donde los indicadores fueron aceptables.

METODOLOGÍA

La primera etapa del estudio considera la definición de indicadores y su medición, el modelamiento de problemas y la implementación de técnicas de resolución y estrategias de enumeración. La segunda etapa considera el diseño, implementación y evaluación de las técnicas de resolución con auto-ajuste/adaptación del proceso de resolución. Acá se integrarán los componentes de la primera etapa, con el fin de que la información dada por la medición de los indicadores permita reaccionar on-the-fly frente a malas elecciones de estrategias de enumeración. Se evalúa el desempeño con instancias benchmark. La estrategia para responder a las preguntas de la investigación, lograr sus objetivos y analizar la certeza de las hipótesis formuladas, corresponderá a la frecuentemente utilizada en Inteligencia Artificial para la resolución de problemas. Aquí, ésta consiste en partir de un problema específico, intentar resolverlo con la técnica de Propagación y Enumeración simple (sin capacidad de adaptación) y en base al análisis de los resultados obtenidos y de su comportamiento, se modificará para que sea capaz de adaptarse y solucionar de mejor manera el problema tratado.

Programación con Restricciones Dinámica

En este trabajo se usó el framework que se muestra en la Figura 3, permitiendo observar lo que está ocurriendo durante el proceso de resolución y decidir cambiar la estrategia de enumeración utilizada cuando se detecta que es ineficiente; esto se logra a través de la interacción de diversos componentes encargados de ejecutar el proceso, capturar información o tomar decisiones de conmutación.

En el framework propuesto se considera una serie de estrategias, cada una asociada a una prioridad, la cual se va actualizando en base a la medición de los indicadores (tamaño del espacio de búsqueda, número de variables instanciadas, número de backtracks). Cuando la estrategia utilizada es ineficiente (se escapa del rango de aceptabilidad), se cambia por aquella de mayor prioridad en ese momento.

ISSN 0717-9103
ISSN Online 0718-8307

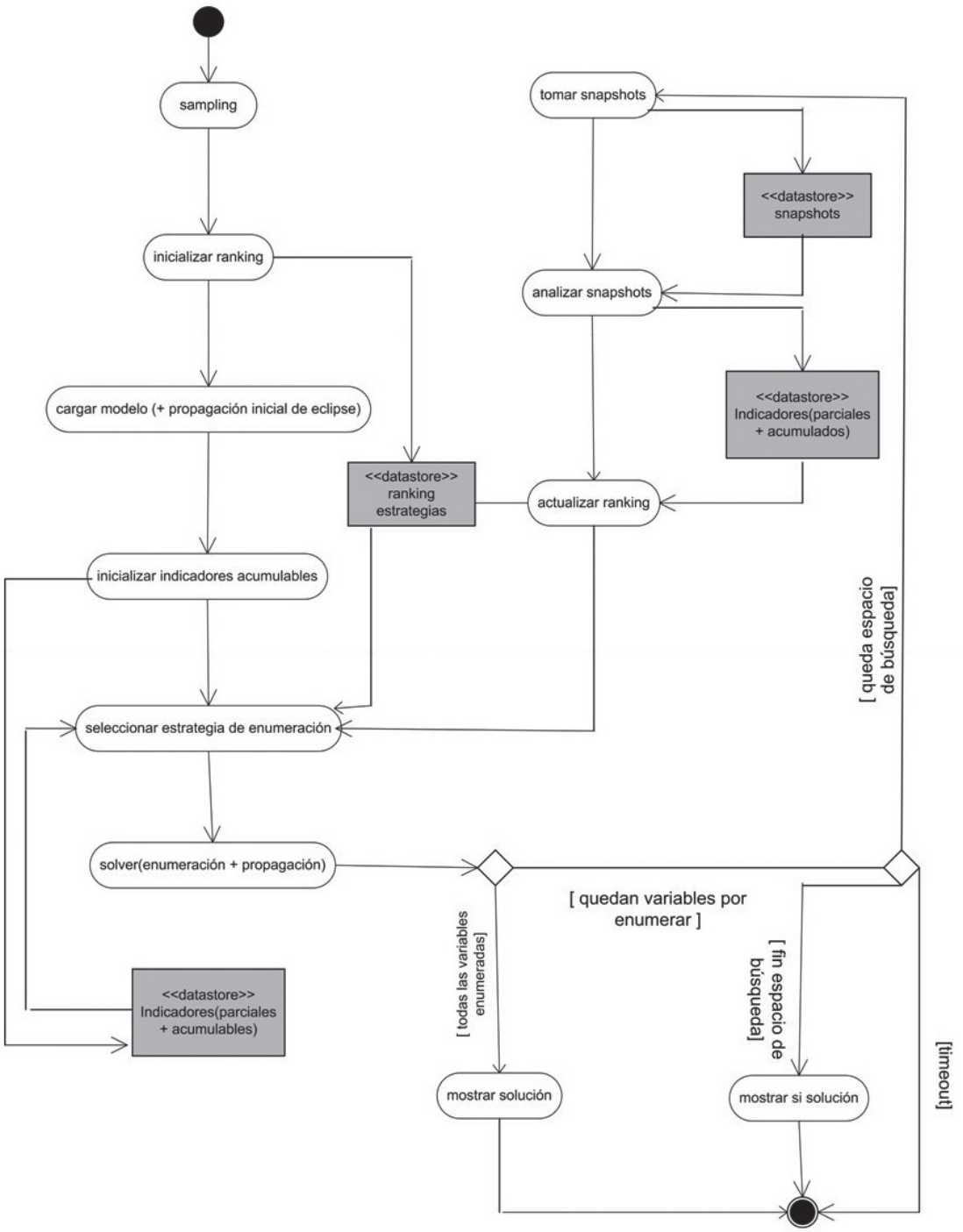

Figura 3: Framework Programación con Restricciones Dinámica

RESULTADOS

Los resultados que se muestran en las tablas 1 y 2 confirman la hipótesis sobre la posibilidad de lograr implementar una estrategia de enumeración adaptativa para CSP que permita entregar en promedio buenos resultados para una variedad de problemas, sin caer en la necesidad de incorporar una estrategia específica para cada tipo de estas.

De esta manera, la estrategia propuesta permite adaptarse dinámicamente a un tipo de problema a resolver, en base a información obtenida experimentalmente y en base a indicadores de procesos que nos permiten verificar en cada momento el estado actual del proceso de búsqueda y, de esta manera, tomar decisiones al respecto. A continuación se presentan algunos de los resultados obtenidos.

Tabla 1: Cantidad de Backtracks realizados por 12 estrategias simples y una adaptativa

	nreinas					c_mágico			Sudoku	
	n=16	n=20	n=25	n=100	n=250	n=3	n=4	n=5	n=3	n=4
First_fail + indomain_min	3	11	21	8	356	0	3	185	0	3
First_fail + indomain_middle	6	5	0	13	3489	1	70	35	0	0
First_fail + indomain_max	3	11	21	8	356	1	97	100000	0	3
First_fail_R + indomain_min	4	52	0	112	25	0	2	132	0	6
First_fail_R + indomain_middle	0	7	0	0	0	0	160	81	0	3
First_fail_R + indomain_max	4	52	0	112	25	0	2	1397	0	6
Input_order + indomain_min	542	10026	2014	100000	100000	0	12	910	28	0
Input_order + indomain_middle	0	13	155	100000	100000	1	24	839	27	0
Input_order + indomain_max	542	10026	2014	100000	100000	1	51	100000	28	0
input_smallest + indomain_min	506	862	885	100000	100000	0	22	100000	100000	0
input_smallest + indomain_middle	3	97	16382	100000	100000	1	172	16525	100000	100000
input_smallest + indomain_max	657	15808	9971	100000	100000	1	95	100000	100000	100000
Estrategia adaptativa	0	7	0	0	0	3	160	81	0	3

Tabla 2: Cantidad de nodos visitados realizados por 12 estrategias simples y una adaptativa

	nreinas					c_mágico			Sudoku	
	n=16	n=20	n=25	n=100	n=250	n=3	n=4	n=5	n=3	n=4
First_fail + indomain_min	20	51	82	122	1345	9	23	546	80	261
First_fail + indomain_middle	26	35	25	145	10341	10	167	108	81	256
First_fail + indomain_max	20	51	82	122	1345	10	230	1000	81	261
First_fail_R + indomain_min	24	128	25	438	340	9	21	371	81	267
First_fail_R + indomain_middle	26	32	25	100	250	9	451	402	81	260
First_fail_R + indomain_max	25	128	25	438	340	9	21	3465	81	267
Input_order + indomain_min	1224	23893	4842	100000	100000	9	37	1901	124	256
Input_order + indomain_middle	16	66	576	100000	100000	10	53	1352	123	256
Input_order + indomain_max	1224	23893	4842	100000	100000	10	10	10000	124	256
input_smallest + indomain_min	970	1706	1495	100000	100000	9	58	10000	100000	256
input_smallest + indomain_middle	22	196	34187	100000	100000	10	246	28572	100000	100000
input_smallest + indomain_max	1513	36401	21516	100000	100000	10	136	10000	100000	100000
Estrategia adaptativa	16	32	25	0	250	9	451	402	81	260

ISSN 0717-9103
ISSN Online 0718-8307

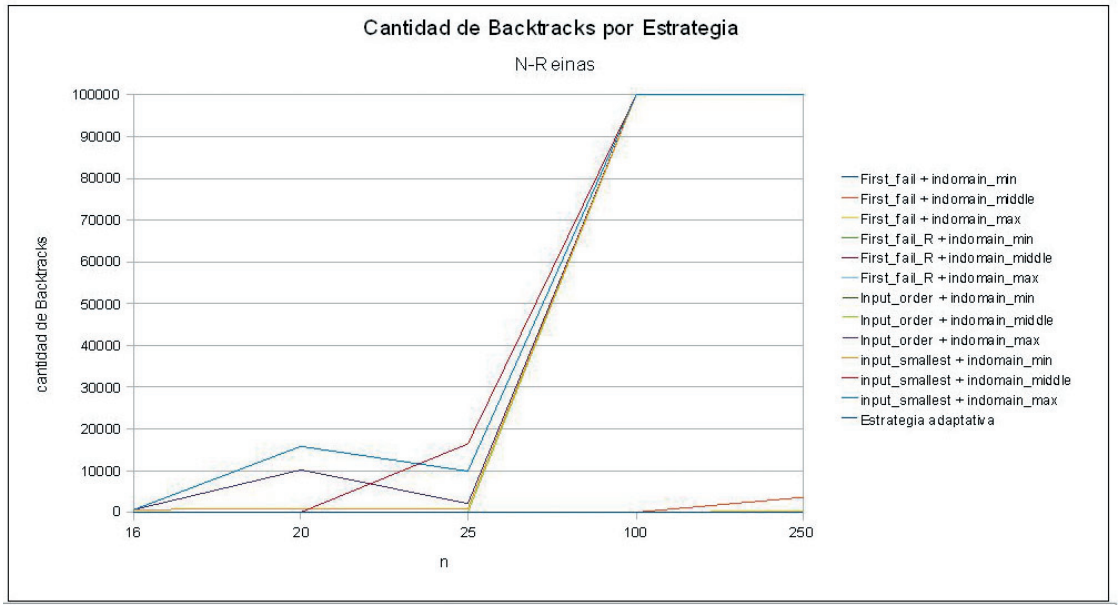

Figura Nº 5: Cantidad de Nodos Visitados para N-Reinas

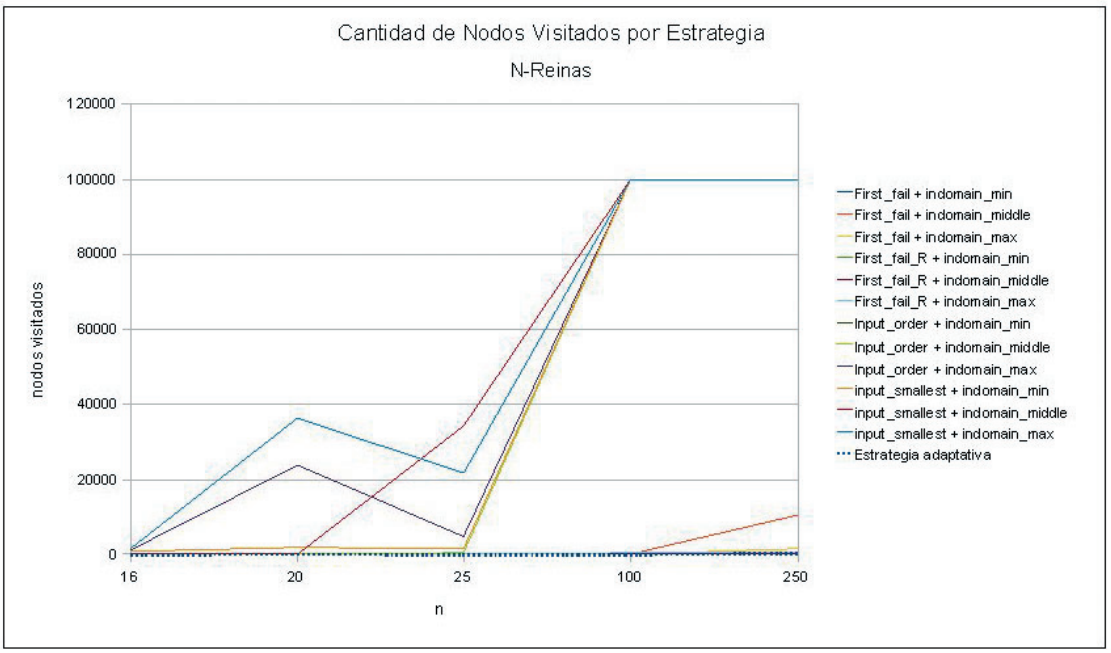

Figura Nº4: Cantidad de Backtracks para N-Reinas

CONCLUSIONES

Se ha implementado una solución del tipo Forward Checking que incorpora diferentes heurísticas de ordenación de variable y valor, y se ha definido una modalidad adaptativa que permite cambiar de estrategia de enumeración al presentarse malos resultados.

En los resultados obtenidos, se ve claramente que la propuesta adaptativa presenta buenos resultados, ya que disminuye la cantidad de backtracks realizados, la cantidad de nodos visitados y, por consiguiente, el costo computacional para la resolución de problemas. Como ya se mencionó, si bien existen heurísticas de ordenación de valor y variable que se comportan relativamente bien para un problema determinado, no se comportan de igual manera cuando son utilizadas para otro problema, ya que aunque estos problemas pertenezcan a la misma clase, tienen restricciones o características propias de su modelamiento que los distinguen de los demás y que, por lo tanto, hacen que sea diferente su resolución, por lo que una propuesta como la planteada en este trabajo se considera de interés y pertinente al propósito de hacer la Programación con Restricciones más usable. Como trabajo futuro se considera el self-tuning de los parámetros considerados (rasgos de aceptabilidad, premio/castigo de prioridades.

REFERENCIAS

Apt, K. (2003). Principles of Constraint Programming. Cambridge University Press.

Barber, F. & Salido, M. (2003). Introducción a la Programación de Restricciones. Revista Iberoamericana de Inteligencia *Artificia*, 20, 13-30.

Barták, R. (1998). On-Line Guide to Constraint Programming. First Edition. http://kti.ms.mff.cuni.cz/~bartak/constraints/index.html

Beasley, J. (1990). OR-Library: distributing test problems by electronic mail, Journal of the Operational Research Society 41, 1069—1072. Disponible en http://people.brunel.ac.uk/~mastjjb/jeb/info.html

Beck, J., Prosser, P., & Wallace,R. (2004). Trying again to fail-first, CSCLP (Boi Faltings, Adrian Petcu, Francois Fages, and Francesca Rossi, eds.), *Lecture Notes in Computer Science* Springer. 3419, 41–55.

Borret, J., Tsang, E. & Walsh, N. (1996). Adaptive constraint satisfaction. In 15th UK Planning and Scheduling Special Interest Group Workshop, Liverpool.

Castro, C., Monfroy, E., Figueroa, C. & Meneses, R. (2005). An approach for dynamic split strategies in constraint solving, MICAI (Alexander F. Gelbukh, Alvaro de Albornoz, and Hugo Terashima-Marín, eds.), Lecture Notes in Computer Science, Springer. 3789, 162–174.

Castro, C., Monfroy, E., Figueroa, C. & Meneses, R. (2005). Constraint Solving Using Dynamic Variable and Value Selection. XXXI Conferencia Latino-Americana de Informática, CLEI 2005, Cali, Colombia

Caseau, Y. & Laburthe, F. (1994). Improved clp scheduling with task intervals, Logic Programming: Proc. of the 11th International Conference on Logic Programming (P. Van Hentenryck, ed.), MIT Press, Cambridge, MA, 369–283.

Harvard Business Norton & Kaplan School (1996). The Balanced Scorecard: Translating Strategy into Action. Harvard Business School Press.

Monfroy, E., Castro, C. & Crawford, B. (2006). Adaptive enumeration strategies and metabacktracks for constraint solving, ADVIS (Tatyana M. Yakhno and Erich J. Neuhold, eds.), Lecture Notes in Computer Science Springer. 4243, 354–363.

Norman M., Sadeh & Mark S. Fox. (1996). Variable and value ordering heuristics for the job shop scheduling constraint satisfaction problem. *Artificial Intelligence,* 86(1), 1–41.

Prosser, P., Christopher, J., Wallace,B. & Richard J. (2003). Toward understanding variable ordering heuristics for constraint satisfaction problems, Fourteenth Irish Artificial Intelligence and CognitiveScienceConference-AICS2003, 11–16.

Schulte, C. & Smolka, G. (2000). Finite Domain Constraint Programming in Oz. A Tutorial.

Sturdy, P. (2003). Learning good variable orderings., CP (Francesca Rossi, ed.), Lecture Notes in Computer Science. Springer. 2833, 997.

Van Roy, P. & Haridi, S. (2004). Concepts, techniques, and models of computer programming, MIT Press.

Zoeteweij, P. (2005). Composing Constraint Solvers. Impreso por PrintPartners Ipskamp, Enschede.