Congreso Internacional en Ingeniería Electrónica. Mem. Electro 2015, Vol. 37, pp 73-80, Chihuahua, Chih. México
http://depi.itchihuahua.edu.mx/display/memorias_electro/MemoriaElectro2015
ISSN 1405-2172

Modelado de Problemas Combinatorios y Dominios Dinámicos. Caso de Estudio: RCPSP en Ambientes de Manufactura

Carmen L. García-Mata [1,2], Pedro R. Márquez-Gutiérrez [2] and Larysa Burtseva [1]

[1] Instituto de Ingeniería, Universidad Autónoma de Baja California,
Calle de la Normal S/N, Col.Insurgentes Este, Mexicali, B.C., C. P. 21280, México
[2] Instituto Tecnológico de Chihuahua
Ave. Tecnológico 2909, Chihuahua, Chih., C.P. 31310, México

{clgarcia,pmarquez}@itchihuahua.edu.mx, burtseva@uabc.edu.mx

RESUMEN.

Este artículo describe el modelado y solución de un problema del tipo combinatorio con Answer Set Programming. Específicamente, el problema resuelto es una instancia del tipo de problemas conocidos como Proyecto del Problema de Planificación con Recursos Restringidos (*Resource Constrained Project Scheduling Problem, RCPSP* . En este ejercicio, se obtiene el schedule sin optimizar y después se contrastan los resultados con el schedule optimizado teniendo como funcion objetivo la minimizacion del tiempo en el cual terminan todos los procesos (min-makespan). Para el tiempo límite empleado en las pruebas se obtuvieron seis soluciones, las cuales se resolvieron en menos de un milisegundo. El programa sin modificaciones en ninguna regla, puede ser empleadas para resolver cualquier otra instancias de dicho problema, solo es necesario actualizar los datos del dominio del problema.

Palabras Clave: Answer Set Programming, calendarización de actividades, RCPSP, Razonamiento no Monotónico Non-monotonic Reasoning.

ABSTRACT.

This paper describes the modeling and solving a combinatorial problem with answer set Programming. Specifically, the solved problem is an instance of the type of problems known as the Restricted Problem with Resource Planning (Resource Constrained Project Scheduling Problem, RCPSP. In this exercise, first the schedule is obtained without optimizing, and then the results are compared with the optimized schedule. Objective of optimization was minimizing the makespan. Program was tested with deadline = 6, and six solutions were obtained in less than one millisecond. The program without any modifications to the rules can be used to resolve any other instances of that problem, it is only necessary to update the data of the problem domain.

Keywords: Answer Set Programming, scheduling, RCPSP, Non-monotonic Reasoning.

1. INTRODUCCIÓN

Los problemas combinatorios permean en prácticamente todas las áreas de investigación. La característica distintiva de esta clase de problemas es que el número de posibles soluciones son finitos pero crecen exponencialmente conforme se incrementa el tamaño de la entrada. Algunos problemas combinatorios típicos son: búsqueda de trayectorias más cortas/baratas, ruteo de paquetes de datos en internet, predicción de estructura de proteínas, planificación y calendarización de recursos , *sheduling.*

En teoría, se puede encontrar la solución óptima de un problema combinatorio dado realizando una búsqueda exhaustiva de todas las posibles combinaciones de las variables que conforman el problema. Desafortunadamente, aunque el número de posibles combinaciones es finito, también es extremadamente grande. En la mayoría de los casos, encontrar una solución óptima que satisfaga un cierto conjunto de condiciones previamente establecido, es un problema intratable, aún para problemas de tamaño relativamente modesto.

Actualmente no existe un método general que nos permita lidiar con toda clase de problemas combinatorios. Comúnmente, se ha adoptado un enfoque híbrido para la solución de este tipo de problemas con diferentes técnicas que van desde Inteligencia Artificial (IA) hasta Investigación de Operaciones. Sin embargo, dada la complejidad del problema,

usualmente se usa una combinación de heurísticas y métodos de búsqueda combinatoria para obtener una solución en tiempo razonable.

Sin embargo, recientemente se han desarrollado un conjunto de poderosos métodos y técnicas del área de Representación del Conocimiento y Razonamiento (*Knowledge Representation and Reasoning*, KRR) que han sido poco explorados para el modelado y solución de problemas de *scheduling* en ambientes de manufactura.

El interés de esta investigación es explorar los métodos antes referidos para el modelado y solución de un problema de *scheduling* del tipo: Proyecto del Problema de Planificación con Recursos Restringidos (*Resource Constrained Project Scheduling Problem, RCPSP*). En particular, se usarán las teorías y metodologías basadas en Lógica No Monotónica (Non-monotonic Reasoning), y un lenguaje de representación del conocimiento basado en dicha lógica, llamado *Answer Set Programming* (ASP).

2. Revisión del Marco Teórico

Puesto que la solución del problema de estudio tiene un carácter interdisciplinario, ya que incide en las áreas de *scheduling*, *rescheduling*, lógica no monotónica, teorías y lenguajes de planificación para ambientes dinámicos, en consecuencia, se hará una breve reseña del estado de arte de las áreas en que se enmarca el presente trabajo.

2.1 Scheduling para Ambientes Estáticos: Esquemas de Notación y Clasificación

Durante largo tiempo se han desarrollado de manera paralela, dos tipos diferentes de notación y esquemas de clasificación para el problema de planificación. Una de estos esquemas proviene del enfoque conocido como *Machine Scheduling (MS)*. El otro, deriva del problema conocido como Proyecto del Problema de Planificación con Recursos Restringidos (*Resource Constrained Project Scheduling Problem, RCPSP*), el cual tiene un enfoque más general que el de *MS*. Una característica distintiva de RCPSP se debe a que los problemas que estudia tienen un sólo producto ó bien pequeños lotes de producción donde hay que calendarizar recursos escasos para actividades con dependencias temporales. Este tipo de situación se da frecuentemente en muchas áreas, incluyendo la de sistemas de manufactura.

El esquema de clasificación *MS* se basa en la notación de Graham et al,(1979), denotado por la tripleta $\alpha|\beta|\gamma$, donde el campo α representa los recursos del ambiente, el campo β se usa para los procesos y las diferentes medidas de eficiencia se denotan en el campo γ. Entre los modelos más comúnes de planificación derivados de la notación de Graham, se encuentran: *Single Machine Scheduling (SMS), Multiprocessor Scheduling (MS), Job Shop Scheduling (JSS),Open Shop Scheduling (OSS), Flow Shop Scheduling (FSS), Hybrid Flow Shop Scheduling (HFS).*

Mientras que el esquema de clasificación de Graham ha sido ampliamente reconocido y utilizado para la clasificación de problemas con el enfoque MS desde hace varias décadas, no ocurre lo mismos con RCPSP, ya que durante bastante tiempo no hubo ningún esquema de clasificación que fuese ampliamente aceptado por esa comunidad de investigación. Los primeros intentos para establecer un esquema de notación se deben a Herroelen et al. (1997). Sin embargo, su esquema no es totalmente compatible con el de Graham, y por consiguiente, no se emplea frecuentemente. Esta omisión fue subsanada por Brucker et al. (1999), quienes propusieron un esquema de notación y clasificación

unificados para RCPSP y MS, mismo que ahora se utiliza ampliamente.

A continuación se resaltará brevemente algunas de las características de este nuevo esquema de clasificación unificado. El esquema $\alpha|\beta|\gamma$ usado en la literatura de planificacion fue extendido de la siguiente forma: el campo α fue extendido para introducir la notación PS (*project scheduling*) y MPS (*multi-mode project scheduling*). Además PS se aumenta a *Psm, σ, ρ, (m* recursos, σ unidades de cada recurso disponible, cada actividad requiere a lo más ρ unidades de los recursos). En el caso de MPS se consideran también recursos renovables y se aumenta su notación a (*MPSm, σ, ρ, μ, τ, ω),* que significa (*multi-mode project scheduling* con *m* recursos renovables, σ unidades de cada recurso disponible, cada actividad requiere a lo máximo ρ unidades de los recursos, μ recursos renovables, τ unidades de cada recurso disponible, y cada actividad requiere a lo mas ω unidades de los recursos). Por lo que respecta al campo β de la clasificación de Graham para problemas del tipo MS, fue extendida para incluir las siguientes características: tiempos de procesamiento (p_j), restricciones generales de precedencia (*prec*), tiempos de procesamiento estocásticos $(p_j = sto)$ fecha límite para duración del proyecto *(d),* relaciones de precedencia entre las actividades (*chains, intree, outtree, tree*) y restricciones temporales generales dadas por tiempos de retraso mínimos y máximos entre actividades (*temp*). Por último el campo de la función objetivo γ, además de las clásicas funciones objetivo como $C_{max}, L_{max}, \sum w_j c_j$ se añadieron funciones para calcular el valor actual neto $\sum c_j^F \beta_j^F$, la nivelación de los recursos $\sum c_j f(r_k(S,t))$, y la inversión de los recursos $\sum c_k max\, r_k(S,t)$

2.2 Lógica No Monotónica y Answer Set Programming

Las dos soluciones semánticas más exitosas y que han servido de base para el desarrollo de muchos de los nuevos lenguajes totalmente declarativos son: la Semántica del Modelo Estable (Gelfond & Lifschitz, 1988) y la Semántica bien Fundada (Van Gelder et al., 1991). Ambas semánticas tienen actualmente numerosas extensiones. Uno de los lenguajes más exitosos basados en esta clase de semántica es ASP, el cual es un lenguaje declarativo basado en la Semántica del Modelo Estable.

ASP fué seleccionado dado que es un lenguaje muy expresivo que permite representar restricciones temporales y de precedencia, razonamiento con conocimiento incompleto y contradictorio. Estas dos últimas, son las características más destacadas de ASP, ya que permite representar conocimiento por defecto (*default*), o sea el razonamiento del "normalmente". La manera en que ASP permite representar y razonar con situaciones del tipo "normalmente", es similar a la forma como los humanos lidiamos con información incompleta utilizando razonamiento de sentido común.

No menos importante en la selección de este lenguaje, es el hecho de que existen eficientes resolvedores (*solvers*) para este lenguaje, los cuáles en conjunto, han sido empleados exitosamente para el modelado y resolución de problemas combinatorios de diferentes áreas que lidian con ambientes dinámicos e información incompleta. Además Clingo, ha ganado varias veces las competencias realizadas para determinar las mejores máquinas de inferencias para lenguajes basados en lógica no monotónica. Clingo ha sido desarrollada por los Laboratorios Potassco de la Universidad de Potsdam, Alemania (Gebser et al., 2010).

La solución de un problema con ASP se obtiene en dos fases: a) en la primer fase se representan el dominio del problema y sus restricciones en ASP, y b) en la segunda fase, una máquina de inferencias encuentra la solución a través de razonamiento no monotónico (Gelfond & Lifschitz, 1988, 1991).

Existen varias maneras de diseñar el modelado de agentes inteligentes con capacidad de adaptación a entornos cambiantes. Sin embargo, estos modelos no involucran razonamiento acerca de los efectos que ocasionan la ejecución de sus acciones. Antes de que un agente ejecute una acción, tiene que tomar en cuenta el estado actual de su contexto. Teóricamente, esto significa que se debe formalizar una noción de causalidad de los potenciales efectos de las acciones para definir las precondiciones necesarias para ejecutar la acción, la relación entre los fluentes (propiedades), así como los efectos directos e indirectos de las acciones.

3. Planteamiento y Modelado del Problema

A continuación se describe cómo se clasificó, modeló y codificó un problema del tipo RCPSP. Un problema muy similar había sido propuesto, analizado y resuelto con ASP por Baral et al.(2001), Baral (2003). El problema en cuestión pertenece al tipo de problemas clasificados como RCPSP y fue seleccionado para esta experimentación porque la mayoría de los problemas de planificación son variantes, extensiones ó restricciones de problemas RCPSP (Brucker et al., 1999).

La aportación de esta investigación para este problema consistió en: a) clasificar el problema de acuerdo a la notación Herrolen-Brucker; b) adecuar algunas de las secciones del código que propusieron los investigadores arriba indicados para que pudiese ejecutarse en la máquina de inferencias Clingo, ya que la versión de dichos autores fue diseñada para resolverse en Smodels; c) una modificación más importante fue la de extender el código para obtener la solución óptima, ya que en la versión original no fue incorporada la opción de optimización.

3.1 Modelado del problema de RCPSP

La descripción de un problema RCPSP puede establecerse de la siguiente forma: Un proyecto consiste de *n+2* actividades. Cada actividad tiene que ser procesada para que el proyecto sea completado. $V=\{0, 1,.., n, n+\}$ denota el conjunto de actividades a planificar (*n* es el número de actividades reales a planificar). Las actividades *j=0*, (*j=n+1*) son actividades ficticias que corresponden al inicio y fin del procesamiento, respectivamente, y no consumen recursos; $R^{\sigma}=\{1, \ldots, k\}$ denota el conjunto de recursos renovables (el número preestablecido de unidades de un recurso está disponible para cada período del horizonte de planificación T). *E* es el conjunto de restricciones temporales ó de precedencia. El tiempo de procesamiento de la actividad *j* se denomina (p_j) y puede ser que $(p_j)\neq(p_i), j\neq i$. *Pred(j)* es el conjunto de predecesores directos de la actividad *j*. (S_j) denota el tiempo de inicio de la actividad *j*, y (C_j) es el tiempo de terminación de la actividad *j*. El período de uso de la actividad *j* del recurso renovable *k* es denotado por (r_{jk}^{σ}). Por lo tanto, $S=\{S_1, \ldots, S_n\}$ es un plan y $C=\{C_1, \ldots, C_n\}$ es el vector de tiempos de terminación. Por último d_j^{max} denota el máximo retraso para que inicie la actividad *j*.

Dadas las características del problema, y de acuerdo a la notación de Brucker et al. (1999), éste problema pertenece a un modelo del tipo

$P|temp|C_{max}$ y es de complejidad *NP-complete* en el sentido amplio.

El dominio de problema de este problema, se representa y está instanciado de la siguiente forma:

Los recursos de RCPSP están dados por $R^{\sigma}=\{r_1, r_2\}$. El conjunto de actividades es $V=\{1,2,3,4_n\}$. La duración de las actividades son $p_1=3,\ p_2=2, p_3=2, p_4=1$.

Las actividades están interrelacionadas por dos clases de restricciones, y el plan debe ajustarse para que cumpla con una fecha límite ó duración del proyecto, *d.* Las restricciones de las actividades son:

1. Restricciones de Precedencia: $i \rightarrow j$, en esta instancia son: $2 \rightarrow 4,\ 3 \rightarrow 1,\ 3 \rightarrow 2$
2. Restricciones acerca del máximo retraso con en el cual debe iniciar la tarea $d_j max$, donde los valores para este problema son: $d_1 max=2,\ d_{2max}=2,\ d_3 max=5,\ d_4 max=3$.

Además, durante el procesamiento cada actividad $j \in J$ requiere $r^{\sigma}_{j,k}$ de unidades de un tipo de recurso $r^{\sigma} \in R$ durante cada instante de tiempo de su duración p_j sin interrupciones. Los recursos tienen capacidad ilimitada. Se asume que los parámetros $r^{\sigma}_{j,k}$ y p_j son enteros, positivos y determinísticos. Las actividades de inicio y fin del proyecto tienen las siguientes condiciones límite $p_n = p_{n+1} = 0$ y $r_{o,k} = r_{n+1}, k = 0$ para todo $r^{\sigma} \in R$. El problema RCPS consiste en encontrar un plan tal que cumpla con las restricciones temporales y de precedencia, y que al mismo tiempo minimice el makespan C_{max}.

3.2 Representación del RCPSP en ASP

A continuación se detalla el dominio del problema, o sea los datos específicos de este problema:

Los recursos del dominio, $R^{\sigma}=\{r_1, r_2\}$, se expresan con:

```
machine(1). machine(2).
```

Las actividades que van a ser procesadas, *V,* el tiempo de procesamiento de dichas actividades $r^{\sigma}_{j,k}$, y las restricciones temporales (*E,* p_j, $d_j max$ y $d_i max$), son representados por

```
job(1..4).
duration(1,3). duration(2,2). duration(3,3).
duration(4,1).
possible_on(1,1). possible_on(1,2).
possible_on(2,2). possible_on(3,1).
possible_on(3,2). possible_on(4,2).
start(1,2). start(2,2). start(3,5). Start(4,3).
```

Las restricciones de precedencia, $Pred_j$, se expresan así

```
depend(2,4). depend(3,1). depend(3,2).
```

En los siguientes párrafos se detallan las reglas generales para cualquier dominio de este tipo de problema.

Para obtener un *answer set* ó solución, en el cual a cada trabajo se le asigne un solo punto de inicio, escribimos:

```
1 {do(J,M,T,D) : machine(M) : time(T) : duration(J,D) } 1 :- job(J).
```

Las restricciones temporales y de recursos se establecen con las reglas:

```
:- do(J1,M1,T1, F1), not
   possible_on(J1,M1).
:- do(J1,M1,T1,F1), start(J1,T2), T2 <= T1.
:- do(J1,M1,T1,F1), do(J2,M2,T2,F2),
   depend(J1,J2), duration(J2,T3),
   T1 <T2 + T3.
```

Las reglas para evitar el traslape de actividades sobre el mismo recurso, son:

```
overlap(T3) :- do(J1,M1,T1,F1),
               do(J2,M1,T2,F2), J1 != J2,
```

```
                T1 <= T2, duration(J1,T3),
                T1 + T3 > T2.
    :- overlap(T3).
```

Las reglas para obtener el plan con el tiempo mínimo de terminación (*makespan*, C_{max}) se detallan a continuación:

Primero, debemos acumular los tiempos de terminación de los trabajos $C=\{C_1, \ldots, C_n\}$ relativos a cada uno de los recursos

```
endJobM1(J1,M1,F1) :-
do(J1,M1,T1,D1),M1 == 1, F1 := T1 + D1.
endJobM2(J2,M2,F2) :-
do(J2,M2,T2,D2),M2 == 2, F2 := T2 + D2.
```

Puesto que el problema puede tener varias soluciones ó *answer sets*, necesitamos encontrar el tiempo de terminación del último trabajo para cada máquina y para cada una de las posibles soluciones

```
chooseEndJM1(J1,M1,B1):-
endJobM1(J1,M1,F1),
endJobM1(J3,M1,F3), M1 == 1, F1 != F3,
J1 != J3,  F1 > F3, B1 := F1.

chooseEndJM2(J2,M2,B2):-
endJobM2(J2,M2,F2),
endJobM2(J4,M2,F4),
M2 == 2, F2 != F4, J2 != J4,
F2 > F4, B2 := F2.
```

Por último, el plan óptimo es áquel que tenga el mínimo makespan:

```
#minimize[chooseEndJM1(J1,M1,B1)= B1].
#minimize[chooseEndJM2(J2,M2,B2)= B2].
```

4. Resultados del Experimento

Para efectos de mostrar todas las soluciones que encuentre el razonador, se eliminaron del programa las reglas relativas a la optimización. El razonador Clingo encuentra todas las posibles soluciones usando la siguiente instrucción:

```
clingo -n 0 -c deadline=6 JSP-domain1C.lp
JSP-schdC14.lp
```

```
Answer: 1
do(4,2,0,1) do(3,1,3,3) do(2,2,1,2)
do(1,1,0,3) endJobM1(1,1,3) endJobM1(3,1,6)
endJobM2(2,2,3)  endJobM2(4,2,1)
chooseEndJM1(3,1,6) chooseEndJM2(2,2,3)

Answer: 2
do(4,2,0,1) do(3,1,4,3) do(2,2,1,2)
do(1,1,1,3) endJobM1(1,1,4) endJobM1(3,1,7)
endJobM2(2,2,3)  endJobM2(4,2,1)
chooseEndJM1(3,1,7) chooseEndJM2(2,2,3)

Answer: 3
do(4,2,0,1) do(3,1,4,3) do(2,2,1,2)
do(1,1,0,3) endJobM1(1,1,3) endJobM1(3,1,7)
endJobM2(2,2,3)  endJobM2(4,2,1)
chooseEndJM1(3,1,7) chooseEndJM2(2,2,3)

Answer: 4
do(4,2,0,1) do(3,2,4,3) do(2,2,1,2)
do(1,1,0,3) endJobM1(1,1,3) endJobM2(2,2,3)
endJobM2(3,2,7)  endJobM2(4,2,1)
chooseEndJM2(3,2,7) chooseEndJM2(2,2,3)

Answer: 5
do(4,2,0,1) do(3,2,4,3) do(2,2,1,2)
do(1,1,1,3) endJobM1(1,1,4) endJobM2(2,2,3)
endJobM2(3,2,7)  endJobM2(4,2,1)
chooseEndJM2(3,2,7) chooseEndJM2(2,2,3)

Answer: 6
do(4,2,0,1) do(3,2,3,3) do(2,2,1,2)
do(1,1,0,3) endJobM1(1,1,3) endJobM2(2,2,3)
endJobM2(3,2,6)  endJobM2(4,2,1)
chooseEndJM2(3,2,6) chooseEndJM2(2,2,3)

SATISFIABLE
Models       : 6
Time         : 0.000
Prepare      : 0.000
Prepro.      : 0.000
Solving      : 0.000
```

Para encontrar la solución óptima, se agregan las reglas de optimización, y usando la misma instrucción que antes, se obtienen los resultados siguientes:

```
Answer: 1
do(4,2,0,1) do(3,1,3,3) do(2,2,1,2)
do(1,1,0,3) endJobM1(1,1,3) endJobM1(3,1,6)
endJobM2(2,2,3)  endJobM2(4,2,1)
chooseEndJM1(3,1,6) chooseEndJM2(2,2,3)
```

Optimization: 3 6
OPTIMUM FOUND
Models : 1
Optimum:yes
Optimization: 3 6
Time : 0.000
Prepare : 0.000
Prepro. : 0.000
Solving : 0.000

En la siguientes tablas se muestran agrupadas las soluciones encontradas para cada máquina. El número dentro de cada celda representa al identificador del trabajo asignado a la máquina y tiempo indicados. La Tabla 1 agrupa las soluciones o *answer sets* para la máquina 1 y las soluciones indicadas en la Tabla 2 corresponden a la máquina 2. El *Answer Set* óptimo, es el número 1, y ha sido destacado en color azul en las tablas de ambas máquinas de trabajo.

Soluciones	t(0)	t(1)	t(2)	t(3)	t(4)	t(5)	t(6)
Answer Set 1	1	1	1	3	3	3	-
Answer Set 2	-	1	1	1	3	3	3
Answer Set 3	1	1	1	-	3	3	3
Answer Set 4	1	1	1	-	-	-	-
Answer Set 5	-	1	1	1	-	-	-
Answer Set 6	1	1	1	-	-	-	-

Tabla 1. Soluciones para Máquina 1

Soluciones	t(0)	t(1)	t(2)	t(3)	t(4)	t(5)	t(6)
Answer Set 1	4	2	2	-	-	-	-
Answer Set 2	4	2	2	-	-	-	-
Answer Set 3	4	2	2	-	-	-	-
Answer Set 4	4	2	2	-	3	3	3
Answer Set 5	4	2	2	-	3	3	3
Answer Set 6	4	2	2	3	3	3	-

Tabla 2. Soluciones para Máquina 2

5. Discusión de Resultados

Las metodologías y técnicas propuestas para la solución de problemas de planificación, ó sea, Razonamiento No Monotónico y Answer Set Programming muestran varias ventajas. La primera ventaja proviene del hecho que la búsqueda de la solución para el problema de planificación se divide en dos partes. De esta forma el investigador se concentra solamente en el modelado y la representación del problema en el lenguaje ASP. La parte de búsqueda de los *answer set* ó solución del problema se realiza de manera automática por la máquina de inferencias. Existen muchos y muy eficientes motores de inferencia que son desarrollados por equipos dedicados y especializados en esta labor. Además, el desarrollo de las máquinas de inferencia no parten de cero en cada caso, sino que continuamente se sigue refinando, mejorando y haciendo más eficiente este *software*. En cambio, cuando se utilizan otros métodos diferentes a éste, para cada aplicación hay que partir de cero para ambas tareas: el modelado y la búsqueda de soluciones; consecuentemente, con el método empleado en esta investigación, se pueden resolver problemas difíciles en menor tiempo.

6. Referencias

1. Baral Ch., Son T.C., Tuan L Ch. Reasoning about Actions in Presence of Resources: Applications to Planning and Scheduling. In Proceedings of International Conference on Information Technology, pp 3-8, Tata McGraw-Hill, 2001.
2. Baral Ch. Knowledge representation, reasoning and declarative problem solving. Cambridge University Press. 2003.
3. Brucker P., Drexl A., Möhring R., Neumann K., Erwing P. Resource-Constrained Project Scheduling: Notation, Classification, Models, and Methods. European Journal of Operational Research 112 (1999) 3 – 41.
4. Gebser M.,Kaminski R., Kaufmann B., Ostrowski M., Schaub T., Thiele S.: A Users Guide to gringo, clasp, clingo, and

iclingo. 2010. potassco.sourceforge.net/labs.html.

5. Gelfond, M, and Lifschitz, V. 1988. The Stable Model Semantics for Logic Programming., IN:Proceedings of ICLP/SLP-88, pp. 1070–1080.
6. Gelfond, M. and Lifschitz, V. 1991. Classical Negation in Logic Programs and Disjunctive Databases, Proceedings of New Generation Computing, pp. 365–386.
7. Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., 1979. Optimization and Approximation in Deterministic Sequencing and Scheduling: A survey. Annals of Discrete Mathematics 5, 287–326.
8. Herroelen W., Demeulemeester E., De Reyck B., A Classification Scheme for Project Scheduling Problems, Technical Report, Department of Applied Economics, Katholieke Universiteit Leuven, 1997.
9. Van Gelder, A., Ross, A., Schlipf, J. 1991. The Well-Founded Semantics for General Logic Programs, Journal of the ACM. Vol. 38, 3. pp. 620–650.
10. Vieira G., Herrman J., Lin E. Rescheduling Manufacturing Systems: A Framework of Strategies, Policias and Methods. Journal of Scheduling 6: 39-62, 2003.