¿CUÁN VIABLES SON LAS PROPUESTAS DE JACK MA PARA LA PLANIFICACIÓN INFORMATIZADA?

Paul Cockshott

CC ciber-comunismo

Resumen: El uso de la optimización lineal para la construcción de planes económicos lo introdujo por primera vez Kantorovich con un algoritmo similar a los desarrollados posteriormente en Occidente por Danzig. En los últimos años los paquetes de optimización lineal se han hecho accesibles en las computadoras de escritorio, un ejemplo es el sistema *lp-solve*. En este trabajo muestro primeramente cómo el paquete *lp-solve* puede ser usado para construir planes macroeconómicos a partir de la información en la tabla *input/output* (*IO*). Luego examino la complejidad computacional empírica del paquete para tratar este tipo de problemas. Esto mostrará que la complejidad es demasiado grande para permitir que el paquete se aplique a tablas *IO* altamente desagregadas. Como alternativa a *lp-solve* explico el algoritmo Armonía y cómo puede extenderse al problema de los planes plurianuales. Se dan mediciones de rendimiento que indican que el algoritmo Armonía tiene una complejidad computacional notablemente menor que *lp-solve*, lo que lo hace más adecuado para planes altamente desagregados.

Palabras clave: *planificación socialista; Kantorovich; complejidad*

El Problema del Plan

Recientemente el emprendedor Jack Ma (fundador de Alibaba) ha recuperado la idea de los planes económicos computarizados (Thornhill 2017), afirmando que con Big Data, redes e informática moderna debería ser posible tener una planificación detallada de la economía china en tiempo real. Para que esto sea factible en una economía tan grande como la china, la complejidad computacional de los cálculos tiene que ser manejable. Las investigaciones en esta materia se remontan a Kantorovich (1965), que definió el problema del plan como la combinación de un número finito de técnicas junto con un vector de recursos predeterminado para maximizar el cumplimiento del objetivo del plan. El objetivo en sí mismo se especificó en términos de un vector que concreta la combinación de productos que se van a elaborar. Por ejemplo, una planta que produce piezas de motor para automóviles podría tener que maximizar la producción suministrada en las proporciones fijas N bloques de motor, 4N pistones más N cigüeñales. Una técnica, en sus términos, era una combinación lineal de recursos que producía una cantidad de producto en proporciones fijas. Con los datos adecuados sobre las ventas de bienes de consumo, que pueden ser proporcionados por plataformas de venta electrónica como Alibaba, junto con las compras estatales planificadas de bienes públicos, todo el funcionamiento de la economía productiva podría definirse en estos términos.

Dada tal especificación del plan, el algoritmo de Kantorovich permitió obtener el mínimo de recursos utilizando una combinación de técnicas. La respuesta proporcionada por su algoritmo eran las intensidades relativas con las que se debía utilizar cada técnica.

Durante la década de 1960, cuando Kantorovich (1960, 1965) trabajaba en el problema de la planificación macroeconómica, los recursos informáticos estaban mucho menos desarrollados que en la actualidad, y no era posible la preparación de planes detallados para toda una economía utilizando sus técnicas.

¿Qué queremos decir con "manejable" en este contexto? Normalmente, tratamos un algoritmo como manejable si es de orden polinómico, pero cuando se trata de conjuntos de datos muy grandes, se requiere una restricción más fuerte para que el tiempo de ejecución sea de un orden polinomial bajo. Lamentablemente, no parece ser el caso de los algoritmos de programación lineal que se derivan de la obra de Kantorovich.

Rendimiento de *Lp-solve* para la planificación económica

Como prueba se elaboró una interfaz (*front end*) de planificación macroeconómica para el ampliamente utilizado paquete de soluciones *lp*. El paquete *lp* permite experimentar con la planificación macroeconómica. Usándolo se pueden calcular planes multianuales para economías de juguete o, en principio, se puede aplicar para calcular planes sectoriales para economías enteras usando tablas de insumo-producto. Está previsto que se ejecute en un entorno Linux, desde la línea de comandos. Toma los datos en forma de hoja de cálculo, lo que permite utilizar las tablas *IO* nacionales publicadas como punto de partida para los estudios de viabilidad.

Se invoca así:

```
java planning.nyearplan flow.csv cap.csv dep.csv targ.csv > plan.lp
```

Las hojas de cálculo se suministran en forma de valor separado por comas (.csv), un formato de intercambio muy utilizado. En primer lugar, se presenta una tabla de entrada y salida de flujos en formato de columna estándar llamada, en el ejemplo, *flows.csv*. Supongamos que contiene los datos que se muestran en la Tabla 1.

Tabla 1. Un ejemplo tabla de flujos Input-Output

partidas	*Hierro*	*Carbón*	*Maíz*	*Pan*
Hierro	0,1	0	0	0
Carbón	2	1	0,04	0,2
Maíz	0	0	1	1
Pan	0	0	0	0
Trabajo	0,3	1	2	0,1
Output	10	5	10	1

Tabla 2. Un ejemplo de tabla de Stock de Capital asociada a la Tabla 1.

partidas	*Hierro*	*Carbón*	*Maíz*	*Pan*
Hierro	10	7	1,4	1
Carbón	2	1	0,04	0,1
Maíz	0	0	1	0
Pan	0	0	0	0

Estamos tratando con una economía simple que produce hierro, carbón, maíz y pan. Este es un formato estándar, aunque diminuto, de tabla de input-output. Leamos hacia abajo la columna de hierro. Al final de la misma se ve la fila de salida. Esto nos dice que al principio la economía produce 10 unidades de hierro, y para hacer ese hierro utiliza 0.1 unidad de hierro existente, y 2 unidades de carbón más 0.3 unidades de mano de obra. De momento no nos preocupemos por la cuestión de las unidades. En principio pueden ser cualquier cosa - toneladas, kilogramos, pies cúbicos, etc. - siempre y cuando cada tipo de producto se mida consistentemente en su propia unidad. Así que se podría medir el hierro en millones de kg, el carbón en millones de libras, el pan en miles de panes, etc., siempre y cuando cada mención de hierro en las tablas sea siempre en millones de kg, cada mención de carbón

en millones de libras, etc. Del mismo modo, para producir 5 unidades de carbón, la economía utiliza 1 unidad de mano de obra y 1 unidad de carbón. Para cada industria hay una columna que describe su fabricación, y cada fila representa los consumos intermedios de un determinado producto.

Todos estos son flujos de productos intermedios. Es decir, flujos que se utilizan inmediatamente como materias primas. Los flujos no incluyen los flujos destinados a reemplazar el stock de capital agotado. Para ello necesitamos dos tablas más. La siguiente es la tabla de *stock* de capital, *cap.csv* (Tabla 2).

Así pues, para generar la producción actual de hierro, necesitamos un stock de capital de 10 unidades de hierro (pensemos en las grandes máquinas) y un stock de reserva de 2 unidades de carbón (pensemos en las pilas de hierro). Estos no son flujos, son las existencias necesarias para continuar la producción. Nótese que no hay una fila de producción o de mano de obra para esta tabla, ya que se supone que la mano de obra y la producción son las mismas que en la tabla *flows.csv*. Sin embargo, las existencias de capital se desgastan. Y algunas existencias se agotan más rápidamente que otras. Una furgoneta, por ejemplo, se deteriora más rápido que una locomotora de ferrocarril. Por lo tanto, un plan a largo plazo necesita saber cuán rápido se desgasta cada tipo de producto. Esto se especifica en la tabla de depreciación *dep.csv* (Tabla 3).

Tabla 3. Un ejemplo de tabla de Depreciación.

partidas	*Hierro*	*Carbón*	*Maíz*	*Pan*
Hierro	0,07	0,07	0,07	0,06
Carbón	0,5	0,5	0,5	0,5
Maíz	0	0	0,5	0
Pan	0	0	0	0

Tabla 4. Una Tabla que especifica los Objetivos del Plan y la Oferta de Trabajo disponible cada año.

partidas	*Hierro*	*Carbón*	*Maíz*	*Pan*
Hierro	0,07	0,07	0,07	0,06
Carbón	0,5	0,5	0,5	0,5
Maíz	0	0	0,5	0
Pan	0	0	0	0

Este cuadro indica el ritmo al que se deprecian cada año las existencias de cada categoría de medios de producción. Así que el 0,07 o 7% del hierro utilizado para hacer hierro se desgasta cada año, mientras que las existencias de semillas de maíz para hacer maíz, se deterioran mucho más rápido, la mitad de ellas se pierden cada año. Finalmente, tenemos una tabla *labtarg.csv*, que especifica las producciones objetivo y la mano de obra disponible cada año del plan (Tabla 4).

En ella se especifican los niveles objetivo de consumo final anual de cada producto. Es importante señalar que no se trata de la producción total, ya que eso también incluiría la producción de materias primas y los bienes de capital de reemplazo. En cambio, especificamos el plan en términos de la producción disponible para el consumo. Así que en el primer año queremos 0,1 unidad de hierro para el consumo directo (cuchillos y tenedores, ollas y sartenes), 3 unidades de carbón para calentar y cocinar, y 2 unidades de pan. La mano de obra

disponible para la economía será 3. El año siguiente la mano de obra ha crecido un 0,01 hasta 3,01 y, con esta mayor mano de obra pretendemos producir un poco más de pan. Y así sucesivamente para cada año. Podemos añadir o quitar años del plan editando la tabla.

El algoritmo de planificación busca maximizar el cumplimiento de estos objetivos del plan para cada año del plan de n años. Lo hacemos generando un programa para el lenguaje *lp-solve* que se imprime en la línea de salida común.

Se compone de una larga serie de desigualdades precedidas de un objetivo de maximización. En este caso se inicia así

```
max:targetFulfillmentForYear1 +targetFulfillmentForYear2
+targetFulfillmentForYear3 +targetFulfillmentForYear4
+targetFulfillmentForYear5;
. . . . . .
```

Se trata de maximizar la suma de los cumplimientos de los planes anuales individuales. Para cada año, el cumplimiento del plan se especifica en términos de cumplir con una proporción determinada de resultados. Dado que el objetivo era producir al menos 0,1 de una unidad de hierro, el estricto cumplimiento del plan no puede ser mayor que 10 veces el consumo final de hierro en el año 1, con reglas similares para el carbón y el pan.

```
targetFulfillmentForYear1 <=10.0 finalConsumptionOfiron1;
targetFulfillmentForYear1 <=0.333 finalConsumptionOfcoal1;
targetFulfillmentForYear1 <=0.5 finalConsumptionOfbread1;
```

Los recursos iniciales se especifican como otras restricciones. Así, la mano de obra disponible tiene que ser ≥ del total de la mano de obra utilizada en cada industria, y ≤ que la oferta pre-dada para el año, 3 en este caso. Se aplican restricciones similares a las existencias preasignadas de bienes de capital. Se asume que estos son capital fijo, es decir, que no pueden ser transferidos entre industrias.

```
labourForYear1>=labourForiron1 +labourForcoal1+labourForcorn1 +labourForbread1;
labourForYear1<= 3.0;
outputOfiron1<=1.0 capitalstockForironMadeUpOfiron1;
outputOfiron1<= 100.0 flowForironOfiron1;
```

Normas adicionales estipulan lo que se debe hacer con la producción:

```
accumulationOfcoal2>=accumulationForironOfcoal2+accumulationForcoalOfcoal2;
+accumulationForcornOfcoal2 +accumulationForbreadOfcoal2;
```

```
productiveConsumptionOfcoal2>=flowForironOfcoal2

+flowForcoalOfcoal2

+flowForcornOfcoal2 +flowForbreadOfcoal2;

finalConsumptionOfcoal2<=outputOfcoal2-accumulationOfcoal2-
productiveConsumptionOfcoal2;
```

Las reglas para la depreciación y la acumulación de capital permiten vincular los planes de los diferentes años entre sí:

```
depreciationIncoalProductionOfcoal2 =0.5 capitalstockForcoalMadeUpOfcoal2;

capitalstockForcoalMadeUpOfcoal2<=capitalstockForcoalMadeUpOfcoal1

+ accumulationForcoalOfcoal1

- depreciationIncoalProductionOfcoal1;
```

Asumiendo que el programa de optimización lineal (Danzig y Wolfe 1961; ;Danzig 2002) ha sido almacenado en el archivo *plan.lp* podemos ejecutar el programa de esta manera:

```
lp_solve <plan.lp—sort >plan.txt
```

La solución completa del plan, se da en un formato bastante detallado y autodescriptivo, que especifica la producción de cada producto cada año, la cantidad de inversión de cada tipo en cada industria cada año, la mano de obra empleada, etc.

La Complejidad de Lp-solve

La complejidad de un problema de planificación dependerá obviamente de la cantidad de productos que tengamos que rastrear o controlar y del período de tiempo en que se ejecute el plan. Para evaluar esto, se realizaron pruebas utilizando modelos de *input-output* de tamaño creciente y número variable de años de horizonte del plan.

Los modelos se obtuvieron sintéticamente mediante generadores de números aleatorios y se estructuraron para garantizar que todos ellos fueran económicamente viables, es decir, que todos los cuadros de *IO* produjeran un producto material excedente. Se sabe que las tablas *IO* crecen en escasez a medida que se desagregan más y más (Reifferscheidt y Cockshott 2014). Dado que el software *nyearplan* aprovecha la escasez, generando sólo restricciones asociadas a las celdas de matriz no cero, es importante que los modelos de prueba muestren el crecimiento de *NlogN* en el número de celdas no-cero que se ha observado en las tablas reales. Así pues, el generador de tablas *IO* aleatorias está configurado para producir tablas con esta propiedad estadística.

Las mediciones muestran que el programa *lp-solve* (Berkelaar, Eikland y Notebaert 2004) tiene un orden de complejidad de N^3, donde N es el número de industrias. Con respecto a los panoramas de los planes, un plan de 2 años con N industrias requiere alrededor de 6,5 años para ser analizado como un plan de un año, un plan de 5 años requiere alrededor de 72 veces más tiempo. La complejidad

general parece ser del orden de N^3 $Y^{2.6}$ con Y como el número de años. Se trata de un polinomio, pero de un orden suficientemente alto como para hacer impracticables los grandes modelos como se muestra en la Tabla 5.

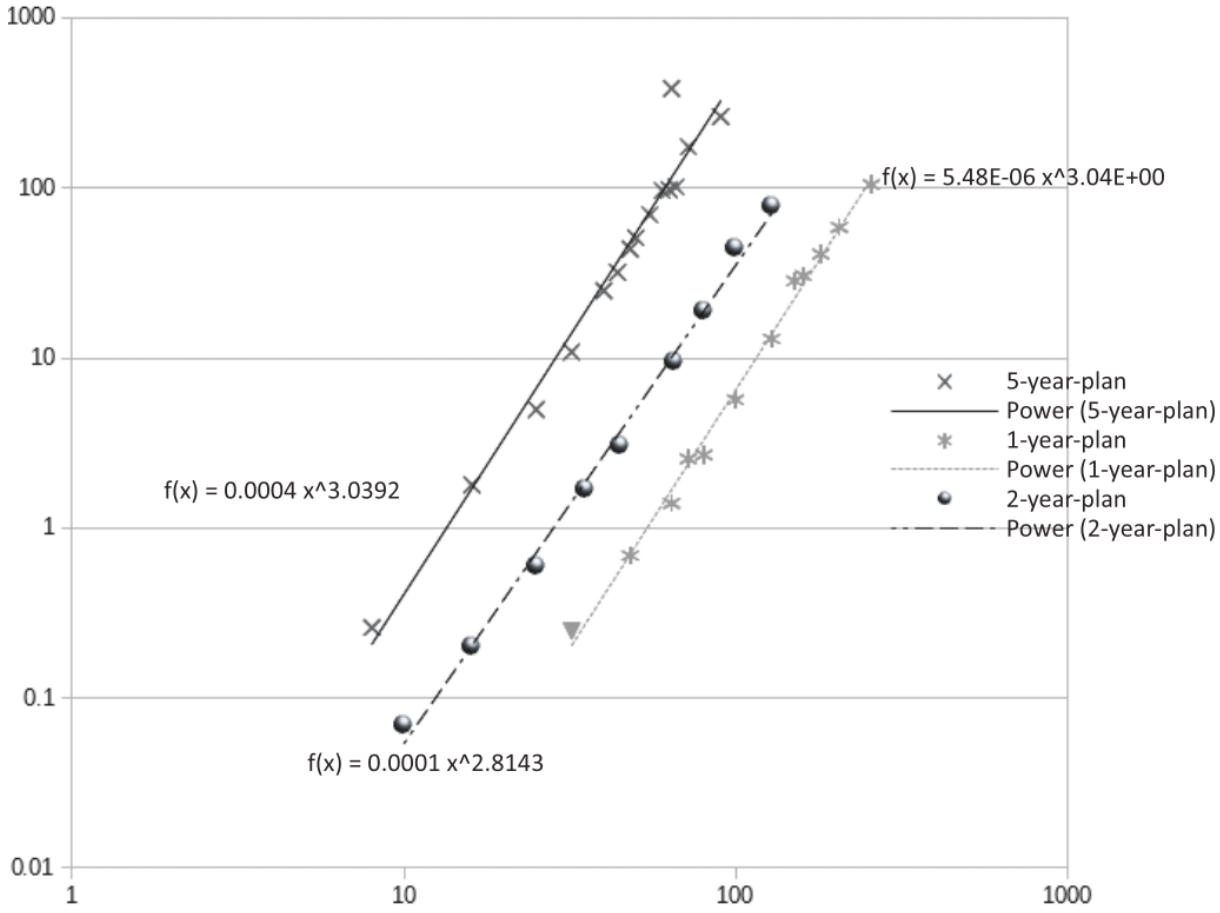

Figura 1. El tiempo de ejecución de *Lp-solve* para la planificación económica parecer ser del orden N^3.

Nota: El rendimiento se midió en un pequeño ordenador Odroid Micro que ejecutaba Ubuntu.

Tabla 5. Tiempo de cálculo proyectado de grandes planes con *Lp-solve*

Industrias	Plan de 1 año	Plan de 5 años
500	0,24 hr	17 hr
5.000	11 días	2,2 años
50.000	33 años	2.417 años

Aunque el solucionador lineal utilizado en *lp-solve* es práctico para la planificación macroeconómica agregada, no lo sería para la planificación detallada. Las pruebas se hicieron en un único núcleo de ARM de 64 bits que funciona a 1,6 Ghz. Un paralelismo como el que existe en los modernos superordenadores permitiría reducir masivamente los tiempos, pero es mucho más preferible empezar con un algoritmo menos complejo. Los economistas soviéticos conocían el rendimiento

relativamente pobre de los optimizadores lineales estándar en planes desagregados. Este resultado presumiblemente se encuentra detrás de los datos citados por el difunto Profesor Nove en su libro (Nove 1983) donde da tiempos astronómicamente largos para que las computadoras construyan planes óptimos para economías enteras. Pero eso no excluye la posibilidad de que haya algoritmos iterativos de baja complejidad que puedan lograr un buen resultado en el mismo problema.

Hay una larga historia de debates entre economistas sobre la viabilidad del cálculo económico planificado (von Mises 1935, 1951; Greenwood 2006; Hayek 1945; Marciszewski 2002; Cockshott y Cottrell 1997; Lange 1938, 1967) en la que una escuela afirma que es factible y otra que es imposible. Ahora bien, cualquier cálculo explícito que pueda ser realizado por humanos también puede ser realizado en principio por computadoras. Además, el orden de complejidad de un algoritmo no cambia dependiendo de si la gente lo hace a mano o si lo hace un ordenador. De ello se desprende entonces que la existencia de economías de mercado en funcionamiento es una prueba de que existen algoritmos de coordinación de baja complejidad. Es evidente que no es cierto que las economías de mercado dependan de un algoritmo de orden N^3 o nunca habrían crecido hasta poder producir los cientos de millones de productos (Gris 2015) que realmente están a la venta. El Algoritmo de Armonía (Cockshott 1990) se basa en ideas de la economía marginalista y de las redes neuronales (Smolensky 1986) para derivar una técnica de planificación iterativa. Se ha demostrado previamente que esto tiene una complejidad *NlogN* para planes de un solo año. Aquí presentamos una implementación adecuada para planes de varios años.

El Algoritmo de Armonía

El algoritmo asume el problema del plan tal como lo definió Kantorovich. En la implementación actual, la interfaz es idéntica a la utilizada en el apartado anterior "Rendimiento de *Lp-solve* para la planificación económica".

Una técnica se define como la producción de cantidades específicas de uno o más *outputs*, utilizando cantidades específicas de uno o más *inputs*, que podemos representar como:

$$P : a_1x_1, a_2x_2, a_3x_3, ... \rightarrow p_1y_1, p_2y_2, p_3y_3, ... \qquad (1)$$

de tal modo que la técnica P utiliza a_1 unidades de input, etc. para producir de output, etc.

Claramente, si sólo usamos una salida, cualquier tabla *IO* de forma Leontief puede fácilmente tener sus columnas representadas con tales técnicas. En el algoritmo se asignan números de índice a los productos y a las entradas para su identificación. Un producto determinado puede aparecer tanto como salida de una técnica como como entrada de otras técnicas, puede ser un recurso pre-dado o un artículo de consumo final que nunca aparece como entrada.

Los períodos de tiempo pueden estar vinculados por las inversiones o por la persistencia de los bienes de capital entre períodos. Una forma de representar la persistencia de los bienes de capital, descrita por Sraffa (1960), es describir cada proceso de producción como la producción de bienes de capital parcialmente utilizados como productos conjuntos. Así pues, una técnica que funcione en un período de tiempo podría producir un producto principal para ser consumido en ese período junto con un conjunto de bienes de capital parcialmente utilizados para estar disponibles en el período. Alternativamente, se puede modelar la inversión en el tiempo como la producción de bienes de capital

disponibles para su uso en períodos en los que *h* es el horizonte de depreciación. En cualquier caso, las técnicas de producción conjunta y, como resultado, durante años tenemos múltiples técnicas alternativas que pueden suministrar los bienes de capital: inversiones en. La combinación de producción conjunta y técnicas múltiples no estaba presente en el algoritmo original publicado. Esto es relevante, porque se han planteado preguntas (Shalizi 2012) sobre si el algoritmo de Armonía es aplicable en estos casos.

La característica fundamental del algoritmo es que, en lugar de especificar una regla estricta según la cual los resultados del plan deben suministrarse en proporciones fijas, se utiliza una función, denominada función de Armonía, para ponderar las desviaciones de la proporcionalidad fija de los resultados.

Se supone que la función de Armonía imita el principio de utilidad marginal positiva pero decreciente. Lo que se requiere es una función cuyo valor aumenta a medida que se acerca el cumplimiento del plan, pero que recompensa el cumplimiento excesivo menos que castiga el incumplimiento. Tal como se aplica, la función es

$$h(t,n) = \begin{cases} S - \frac{S^2}{2} & S < 0 \\ \ln(S+1) & S \geq 0 \end{cases} \qquad (2)$$

donde

$$S = \frac{n-t}{t} \qquad (3)$$

y *t* es el objetivo del plan, y, *n*, la producción neta del producto correspondiente. La forma de la función se muestra en la figura 2.

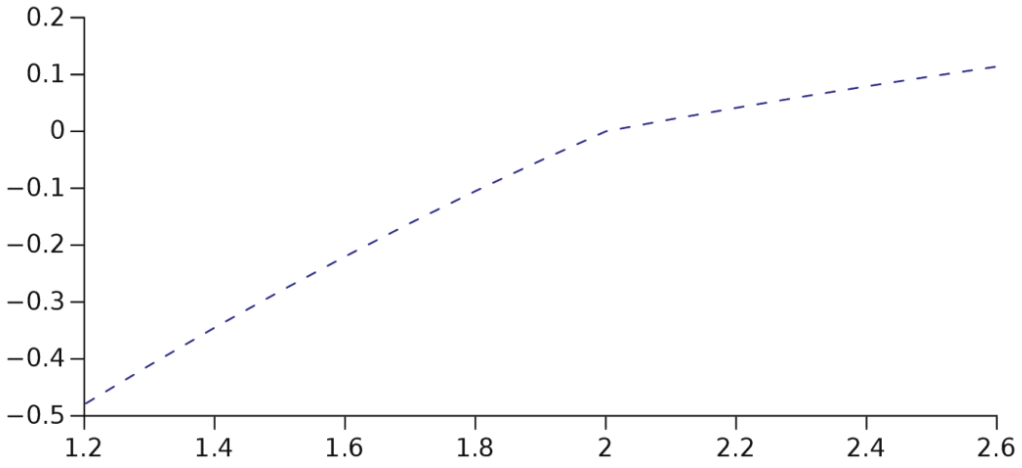

Figura 2. La función de armonía para un *Output* planificado de 2 unidades.

Nota: Armonía = 0 cuando el plan cumple el objetivo.

Es continua, tiene una primera derivada positiva y una segunda derivada negativa. Dado un

complejo de tecnologías *C*, un plan ray *P*, un vector de recursos iniciales *R* y un vector de intensidad inicial *I*, especificando la escala de producción de cada una de las técnicas, es sencillo calcular el vector de salida neta *N* y a partir de éste se obtienen el vector de armonía *H* y el vector de gradiente de armonía *∇H*. En el caso de los bienes no finales, el derivado de armonía se obtiene de la media de los derivados de armonía de los bienes a los que contribuyen multiplicados por su producto físico marginal en cada contexto.

$$\delta h_{\mathrm{nf}}(i) = \mu(\frac{dh}{dy_j} mpp_{j,i,p} \forall j : \mathrm{uses}(i,j,p) \forall p) : i \in \mathrm{nf} \qquad (4)$$

donde nf es el conjunto de bienes no finales, es el producto físico marginal del bien *j*-ésimo en términos del bien *i*-ésimo en la técnica *p*-ésima, y el uso de *(i,j,p)* es un predicado que es cierto si la técnica *p* usa *i* en la producción de *j*.

Para cada técnica de producción *p* se obtiene una tasa de ganancia de armonía por ciclo de producción, que viene dada por

$$G_p = \frac{\sum p_i dh/dy_i - \sum a_j dh/dx_j}{\sum a_j dh/dx_j} \qquad (5)$$

el hecho de que la a_i y la p_i son los coeficientes de producción y utilización en la ecuación 1. Esta tasa de ganancia de armonía por ciclo es un análogo computacional de la tasa de rendimiento del capital.

El algoritmo itera sobre tres pasos:

1. Una fase de Newton Raphson que estima la armonía media y trata de mover la escala de producción de todas las industrias hacia el nivel en el que estarían operando en armonía media. Las industrias *i* se expanden mientras que las otras se reducen. Las intercepciones con la media se determinan usando los gradientes ∇H y un factor de escala $\psi<1$ se usa para desplazar las industrias parcialmente hacia la media. El ajuste de cualquier industria tendrá efectos de segundo orden en términos del cumplimiento del plan de materias primas y productos intermedios que utiliza.

2. Una fase de escalada [*hill creep phase*] análoga al movimiento de capital en una economía de mercado hacia las ramas donde la tasa de rendimiento es más alta. La intensidad I de cada técnica j se ajusta de tal manera que donde $\phi<1$ es otra constante de control de la tasa y $\sigma(x)$ es una función sigmoide para mapear -∞..∞ en el rango -1..1. Una función sigmoide adecuada es

$$\sigma(x) = \begin{cases} \frac{x}{1+x} & x > 0 \\ 0 & x = 0 \\ -\sigma(-x) & x < 0 \end{cases} \qquad (6)$$

3. Una fase final de ajuste de consistencia para asegurarse de que no hay un producto neto negativo de ningún tipo. Para ello se busca en el vector de los productos netos los valores negativos. Si se encuentra

alguno, se calcula la escala relativa de reducción de uso que sería necesaria para eliminar el déficit. Todas las técnicas que utilizan el producto están entonces programadas para reducirse por lo menos en esta cantidad. Si una técnica está programada para ser reducida en un 5% debido a la escasez de carbón y un 4% debido a la escasez de hierro, entonces la reducción total será del 5% y no del 9%.

El primer paso produce una convergencia de todos los productos hacia la armonía media. Pero debido a que el gradiente de la función de armonía está disminuyendo, si se intenta mover la producción hacia la intersección por alguna fracción ψ de la distancia a la media, las industrias con una armonía superior a la media se reducirán en una fracción mayor de su producción que la correspondiente expansión de las que están por debajo de la armonía media. El efecto neto del paso 1 es, por lo tanto, reducir la dispersión alrededor de μh mientras se reduce la propia μh. Esto también da como resultado una capacidad no utilizada. El paso 2 es necesario para hacer que la capacidad no utilizada sea absorbida. El paso 2 tiende a ser expansivo.

Podemos asumir que el complejo tecnológico es factible, es decir, que es capaz de producir una salida física neta positiva. Así, aunque para ciertos vectores ∇H, una minoría de industrias puede tener un resultado negativo, esto será compensado por otras con un resultado positivo. El impacto general de un aumento de la intensidad media de la producción utilizando recursos ociosos debe ser un aumento de la armonía, de modo que el efecto neto de la fase 2 es trasladar los recursos ociosos a la producción y aumentarlos. En el proceso, por efectos de segundo orden del consumo productivo, aumentará ligeramente la dispersión en torno a μh. La siguiente iteración de la fase 1 tiende entonces a anular la dispersión, etc.

Inicialización

La intensidad de todos los procesos se inicializa al 20% y las existencias de capital inicial para cada matriz de capital no nula de cada año se derivan de las existencias iniciales debidamente depreciadas. Para cada año distinto del último, se define una técnica de acumulación que consume una unidad del tipo de capital apropiado en el año *t* y entrega una unidad del tipo de capital apropiado, debidamente depreciada, cada año subsiguiente. Al igual que con el solucionador de programación lineal descrito anteriormente, se supone que las existencias de capital son fijas y no se pueden redistribuir fuera del sector en el que se invirtieron originalmente.

Resultados

Tanto para el *lp-solve* como para el algoritmo de planificación basado en la armonía, los tiempos excluyen la lectura inicial y el análisis de las tablas *IO*. El solucionador de Armonía es tan rápido que el tiempo de cálculo se ve completamente disminuido por el tiempo de *IO* en los ejemplos de mayor tamaño.

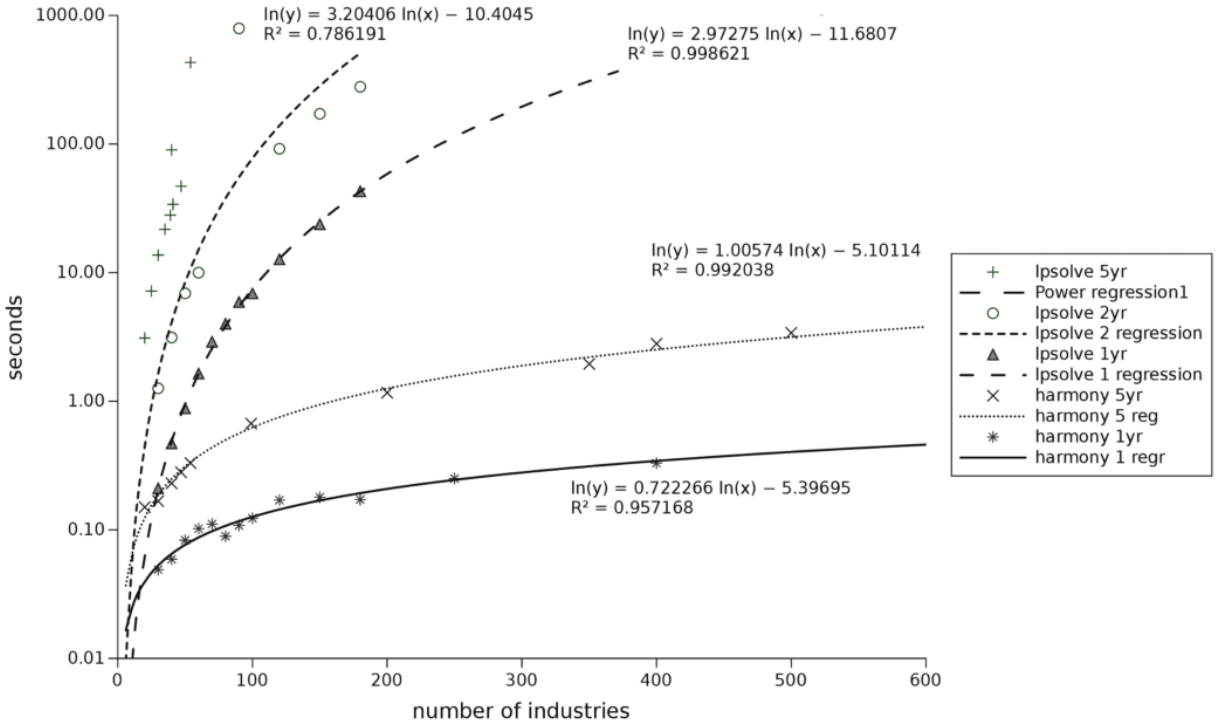

Figura 3. Comparación del rendimiento de los solucionadores de planes basados en *Lp-solve* y Armonía.

Nota: El eje *y* es una escala logarítmica. Pruebas realizadas en un AMD A12 de un núcleo a 1 Ghz usando Ubuntu bajo Virtual Box con 2 núcleos disponibles.

Los resultados se muestran en la figura 3. Está claro que en el rango de planes testados, el *optimizador* de armonía es mucho más rápido que la versión *lp-solve*. En los gráficos que se muestran, parece que el solucionador de Armonía es, durante un número determinado de años, aproximadamente lineal en el número de industrias que se están planificando. Más plausiblemente, es de orden *NlogN*, pero esto es difícil de distinguir dado el rango del conjunto de datos.

El formato de los datos de entrada, utilizando matrices *IO*, es muy ineficiente para un gran número de industrias. El espacio que ocupan las tablas de planificación y los resultados impresos crece según el orden. En la actualidad esto limita el número de industrias con las que se trabaja a menos de 1000 para fines prácticos. Por encima de ese nivel uno se queda sin espacio en el módulo java que analiza las hojas de cálculo. Esto se debe a la ineficiente utilización del espacio de memoria por el sistema de análisis de hojas de cálculo de Java.

Desde el punto de vista del algoritmo de planificación subyacente, eso no es un problema. Dados los datos proporcionados de forma más compacta, por ejemplo de bases de datos relacionales, el espacio no sería un problema hasta que se encuentre un número mucho mayor de productos o industrias, ya que el almacenamiento utilizado para el cálculo real crece sólo en proporción a las entradas no nulas de la matriz de producción.

La aplicación a gran escala se vería favorecida por un paralelismo masivo, pero el algoritmo se presta muy bien a ello. Cada uno de los pasos del algoritmo principal puede ejecutarse en modo de datos paralelos, siempre que todos los núcleos sincronicen el acceso a los vectores compartidos *I*, *H*,∇H entre los pasos. El rendimiento del algoritmo de Armonía se refleja en planes más grandes usando un multiprocesador en la Tabla 6, que se compara muy favorablemente con los 2.540 años de la Tabla 5.

Tabla 6. Tiempo estimado para computar planes de 5 años mediante el algoritmo de armonía.

Productos	*1 hilo*	*1000 hilos*
50.000	9,4 hr	
50.000.000	94 hr	5,6 mins
200.000.000	377 hr	22 mins

En general, llegamos a la conclusión de que, dados los recursos de que disponen las industrias informáticas chinas, los objetivos planteados por el Sr. Ma son, en efecto, técnicamente factibles.

Notas

Tanto el paquete *lp-solve* como el,algo mas complejo paquete de armonía están disponibles en: https://github.com/wc22m/5yearplan.

REFERENCIAS

Berkelaar, M., K. Eikland, and P. Notebaert. 2004. “Lp Solve: (Mixed Integer) Linear Programming Problem Solver.” Eindhoven University of Technology Working Paper No. 63.

Cockshott, W. P. 1990. “Application of Artificial Intelligence Techniques to Economic Planning.” *Future Computing Systems 2* (4): 429–443.

Cockshott, P., and A. Cottrell. 1997. “Information and Economics: A Critique of Hayek.” *Research in Political Economy* 18 (1): 177–202.

Dantzig, G. B., and P. Wolfe. 1961. “The Decomposition Algorithm for Linear Programming.” Econometrica 29 (4): 767–778.

Dantzig, G. 2002. “Linear Programming.” *Operations Research* 50 (1): 42–47.

Greenwood. D. 2006. “Commensurability and Beyond: From Mises and Neurath to the Future of the Socialist Calculation Debate.” *Economy and Society* 35 (1): 65–90.

Grey, P. 2015. “How Many Products Does Amazon Sell?” https://export-x.com/2015/12/11/how-many-products-does-amazon-sell-2015.

Hayek, F. A. 1945. “The Use of Knowledge in Society.” *American Economic Review* 35 (4): 519–530.

Kantorovich, L. V. 1960. “Mathematical Methods of Organizing and Planning Production.”

Management Science 6 (4): 366–422.

Kantorovich, L.V. 1965. *The Best Use of Economic Resources*. Cambridge, MA: Harvard University Press.

Lange, O. 1938. *On the Economic Theory of Socialism*. Minneapolis, MN: University of Minnesota Press.

Lange, O. 1967. "The Computer and the Market." In *Socialism, Capitalism and Economic Growth: Essays Presented to Maurice Dobb*, edited by C. H. Feinstein, 158–161. Cambridge: Cambridge University Press.

Marciszewski, W. 2002. "Hypercomputational vs. Computational Complexity: A Challenge for Methodology of the Social Sciences." *Studies in Logic, Grammar and Rhetoric* 5 (18): 11–31.