# INTENT Un Algoritmo Interior para Programación Entera

*Rafael A. Pastoriza*

Escuela de Computación
Facultad de Ciencias, Universidad Central de Venezuela
Apdo 47002, Caracas, Venezuela
rpastoriza@cantv.net

---

**Abstract**

*Some ideas are presented that suggest a heuristic to solve integer linear programming problems* **ILP**. *Interior points are successively explored inside a convex of a* $\mathbf{LP}(e)$ *problem. The latter linear programming problem has a convex, that contains the original integer solutions, at a fixed distance* $e > 0$ *from the convex defined by the problem restrictions, and is obtained relaxing the integer condition on variables. Within the convex cone with origin in the* $\mathbf{LP}(e)$ *optimal finite real solution, an interior polyhedron is constructed at certain distance from vertex, with extreme points in the convex cone edges. Inside that polyhedron are sampled points, and also inside a cube with origin in those points. Points, that rounded give integer vectors. The* **ILP** *feasibility is checked. The interior polyhedrons are recalculated, inside the convex cone, increasing its distance from the real optimum. If a feasible integer solution is founded, then the searching path is retraced looking for a better point. Using* **INTENT** *the solution for some test problems are been obtained.*

**Resumen**

*Se presentan algunas de las ideas que llevaron a sistematizar e implantar un algoritmo heurístico para resolver problemas de programación lineal entera* **PPLE**. *Esto se hace explorando puntos interiores a un* $\mathbf{PPL}(e)$, *el cual es un problema de programación lineal, que se obtiene relajando en el problema original la condición de integridad en las variables, y cuyo convexo, separado a una distancia fija* $e > 0$ *del convexo dado por las restricciones del* **PPLE**, *contiene las soluciones enteras del problema. Se construye un poliedro interior al cono convexo con vértice en la solución real óptima del* $\mathbf{PPL}(e)$, *a cierta distancia del vértice, cuyos puntos extremos están en las aristas del cono. Se muestrean puntos al azar dentro de este poliedro. Además, se muestrean puntos interiores a cubos que tienen como origen de coordenadas a los anteriores. Puntos que, al ser truncados, proporcionan vectores enteros, a los cuales se les calcula su factibilidad en el* **PPLE**. *El poliedro interior se recalcula sucesivamente cada vez alejándose a mayor distancia del óptimo real, vértice del* $\mathbf{PPL}(e)$, *en dirección interior del cono convexo, para obtener soluciones enteras al* **PPLE**. *Si se encuentra una solución entera factible entonces la búsqueda se devuelve hacia el óptimo*

*real buscando una solución mejor. En general, y de acuerdo a los valores que tengan ciertos parámetros propios de* **INTENT**, *se logran obtener las soluciones óptimas conocidas para algunos problemas de prueba conocidos.*

**Keywords:** Integer Programming, Heuristics.

## 1 Introducción

Un problema de programación lineal entera se puede escribir como:

$$\begin{array}{c} Max \quad cx \\ s.a. \\ Ax \leq b \ (\mathbf{PPLE}) \\ x > 0 \ x \quad entero \end{array}$$

En Pastoriza [1] y [2] aparecen expuestas algunas de las ideas que sugieren como puede ser desarrollado un algoritmo interior heurístico para resolver problema de programación entera. Descansan en la siguiente observación; quizás, salvo patologías extremas, el óptimo en un **PPLE** no debe estar sustancialmente lejos de la solución de ciertos problemas de programación lineal **PPL**. Por caso aquellos que se obtienen relajando la condición de integridad en las variables del problema entero original. O en otros términos, se trata de explotar la siguiente idea. Desde el óptimo real de ciertos problemas lineales cercanos al problema original se debe poder visualizar la solución entera óptima, buscando en sentido interior al cono que ahí se origina, y, por consiguiente, debe ser posible alcanzar el óptimo entero mediante algún artificio; por supuesto, en aquellos casos en los cuales exista tal solución óptima entera.

Un poco de reflexión muestra que si el punto entero buscado, es interior al convexo naturalmente asociado al **PPLE**, éste se puede llegar a conseguir truncando algunos valores reales que lo mayoren. Si el punto entero es interior al convexo, existe un entorno de puntos reales interiores que lo contiene, entre los cuales algunos lo mayorarán y por ende, el entero buscado ser alcanzable por truncación. Sin embargo, para aquellos problemas en los cuales el punto entero está sobre la frontera del poliedro convexo, su obtención se puede llegar a escapar. En estos casos no existe una bola abierta, con centro en él, que esté totalmente contenida dentro del convexo; y tal vez entonces el punto entero buscado será sólo alcanzable truncando puntos externos.

Para sortear la observación anterior basta hacer la truncación sugerida pero desde puntos interiores a un poliedro convexo mayor que contenga al original; de manera tal que todo punto entero factible al **PPLE** sea interior al nuevo poliedro. O sea, se trata primero de ponerle una camisa a una cierta distancia positiva $e$, *espesor*, al **PPL** ahora relajado y aumentado. Este **PPL**$(e)$, que se obtiene relajando en el

**PPLE** la condición de integridad y agrandándolo de manera homogénea, se denomina "*camisa*". En el apéndice 1 aparecen las correspondientes expresiones. Resolviéndolo, conociendo los valores de su solución óptima real, y suponiendo que exista solución finita, se encuentran las aristas que emanan del punto óptimo real $\mathbf{x_B^*}(e)$. Sus vectores directores normalizados $\delta_i$indicarán las direcciones de las aristas $v_i$ del cono convexo, dentro del cual, en caso de existir, debe estar contenido el óptimo entero que se busca.

Utilizando estas aristas $v_i$ se puede construir un poliedro plano interior al cono, dirigido por el vector funcional $c$, con puntos extremos $P_i$ a una cierta distancia $d$ preestablecida de la solución lineal $x_B^*(e)$. Estos puntos $P_i$ estarán entonces, por construcción, sobre cada una de las aristas. Muestreando al azar valores en ese poliedro convexo se obtienen puntos $x_i$ reales que en algunos casos mayoran a las soluciones del **PPLE** original; más aún incluso se los puede emplear como origen de coordenadas de un cubo con una cierta magnitud preestablecida en sus lados, y explorar entonces su vecindad, muestreando al azar puntos $y_j$ dentro de ese cubo. Truncando los $y_j$ así obtenidos se obtienen puntos enteros $z_j$ candidatos a proporcionar el óptimo entero del **PPLE**. Los vectores truncados $z_j$ resultantes pueden haber sido ya obtenidos, puede ser infactibles, o pueden, siendo factibles y distintos, no mejorar el valor óptimo actual; todos aspectos que se deben controlar de manera cuidadosa y debidamente al implantar un algoritmo como el sugerido.

Para detener el algoritmo basta considerar, por ejemplo, que superada una cierta distancia del óptimo real de la camisa al hiperplano que contiene al poliedro convexo, puede ya no estar el óptimo entero buscado.

En la Fig.1 se ilustra la información relevante a la aplicación del algoritmo en $R^3$. Las $z$ indican los puntos enteros; $x$ el punto muestreado dentro del poliedro convexo con puntos extremos $P_1P_2P_3$; $y$ son los puntos muestreados dentro del cubo con centro en $x$; $e$ es el espesor que define la camisa y el problema de programación lineal **PPL($e$)** asociado al **PPLE** dado; y con $x_B^*(e)$ se indica la solución óptima finita real. Los $\delta_i$ son los vectores unitarios que dirigen las aristas del cono desde el vértice $x_B^*(e)$. Es evidente que pueden aparecer durante el proceso esbozado, tanto puntos **PPLE** factibles como no, y que el mismo punto entero puede aparecer varias veces al truncar distintos puntos $y_i$.

Claro está que en la medida en que se muestreen pocos puntos, o el paso de separación entre los hiperplanos se haga muy grande, o el muestreo se haga en un cubo muy pequeño, la solución al **PPLE** puede llegar a escapar al procedimiento descripto.

Tratando de mejorar el procedimiento recién descripto, la búsqueda del óptimo puede ser realizada en dos direcciones. Una *descendiente*, alejándose del óptimo real finito $x_B^*(e)$, en sentido interior, hasta que se encuentre algún punto entero o bien se detenga la búsqueda cuando el hiperplano está demasiado lejos del vértice $x_B^*(e)$. El otro sentido es *ascendente*. Si un punto entero factible ha sido alcanzado mien-

tras se desciende, entonces la búsqueda se puede revertir, hasta volver al óptimo real. Posiblemente convenga en estas circunstancias refinar el paso de la búsqueda y el tamaño del muestreo, buscando encontrar así un punto entero mejor que el actual.

Es conveniente considerar en la implantación alguna clase de rechazo para mejorar la eficiencia computacional del procedimiento esbozado.

La experimentación con el algoritmo puede llegar a contestar valores juiciosos para los valores que especifican sin ambigüedades el algoritmo, valores por ahora desconocidos.

- El algoritmo bosquejado se denomina **INTENT**, como una contracción de *interior entero*.

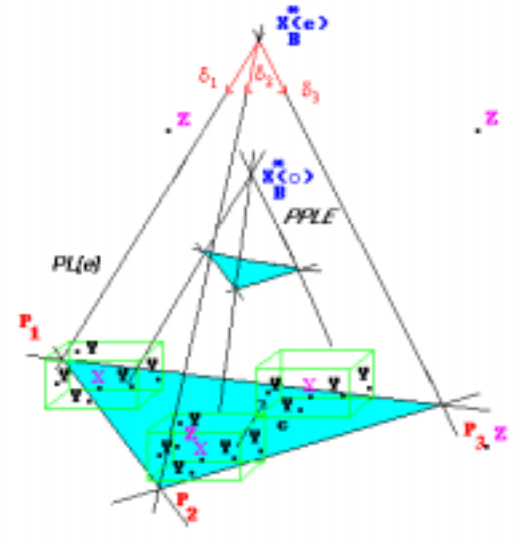


Fig. 1: Algoritmo **INTENT**

## 2 Algoritmo INTENT

Problema entero.

Dado un problema de programación entera:

$$\begin{array}{c} Max \quad cx \\ s.a. \\ Ax \leq b \ (\mathbf{PPLE}) \\ x \leq 0 \quad x \quad entero \end{array}$$

**1. Resolver el PPL($e$)**

Ingresar $e \geq 0$, valor del espesor y resolver el **PPL($e$),** problema de programación lineal con una camisa de espesor $e$.

$$\begin{array}{c} Max \quad cy \\ s.a. \\ Ay \leq b(e) \ (\mathbf{PPL}(e)) \\ y \geq 0 \\ donde \quad y = x + e1, \quad b(e) = b + e\,sa + e\,an \end{array}$$

Aquí 1 es un vector con un (1) uno en cada componente; $an$ un vector cuyas componentes son la norma euclidiana del vector $a_i$, $i$-ésima fila de la matriz $A$; y $sa$ un vector cuyas componentes son las suma de los valores de las filas de la matriz $A$. Todos los vectores con las dimensiones apropiadas.

Sea $zoptreal = cy^*$ el valor del funcional en $y^*$, solución real óptima finita de este problema.

Sean $ar_1, ar_2, ..., ar_n$ las aristas del cono convexo del **PPL($e$)** que tiene por origen el punto óptimo real $y^*$.

Calcular los vectores directores $\delta_i$ de las aristas $v_i$.

**2. Inicializar valores INTENT**

$solución \quad óptima := vector \quad entero \quad infactible;$
$funcional \quad óptimo := -\infty;$
$d := 0.0;$
$separación :=\mid \alpha\, zoptreal - funcional \quad óptimo \mid;$
$\alpha := 1.5$

**Entrada:**

paso de búsqueda($deltin$); tamaño muestreo en poliedro ($Npol$); extremo inferior del cubo($lmin$); extremo superior del cubo($lmax$); tamaño del muestreo cubo($Ncubo$); semilla($seed$).

## 3. Búsqueda INTENT

Iterar en la distancia $d$, desde el hiperplano actual a la solución óptima real $y^*$, decreciendo con pasos $deltin$, mientras la distancia $d$ sea menor que la *separación* o que $\mid zoptreal \mid$.

En caso de encontrar una solución entera factible cambiar el signo de $deltin$.

**3.1.** Fijado un valor positivo de distancia $d$.

**3.1.1.** Calcular los coeficientes $\lambda_i$del hiperplano actual que está a esa distancia $d$ y se apoya en las aristas $ar_1, ar_2, ..., ar_n$.

Calcular los puntos extremos $P_i$ del poliedro.

**3.1.2.** Iterar en el tamaño de la muestra en el poliedro ($Npol$).

En el poliedro, muestrear al azar puntos $x$, combinación convexa de los puntos extremos $P_i$.

**3.1.3.** Iterar en el tamaño de la muestra en el cubo ($Ncubo$).

En el cubo, muestrear al azar puntos $y$ , con lados $[lmin, lmax]$, y origen en $x$.

Truncar el punto $y$ dando el punto $z$. Usar un método de rechazo.

Verificar que $z$ sea diferente del óptimo actual.

Verificar que $z$ sea factible al **PPLE.**

Verificar que $cz$ sea mejor que el óptimo actual, quedándose con $z$ en caso afirmativo.

Actualizar el valor de $separación := \mid \alpha \; zoptreal - funcional \quad óptimo \mid$

Fin de iteración en el muestreo del cubo

Fin de iteración en el muestreo del poliedro

Fin de iteración en la distancia

**Salida:**

Solución óptima entera, valor funcional óptimo

## 3 Implantación

El algoritmo **INTENT** bosquejado en el punto anterior debe ser especificado de tal manera que a la vez sea práctica y clara la secuencia de cálculos bosquejados.

**a)** Ponerle "*camisa*" al **PPLE** de manera homogénea, a una misma distancia $e$ positiva, a todo el poliedro convexo definido por las restricciones $Ax \leq b, x \geq 0$, no ofrece dificultad. Se deben cambiar las variables por un lado, y alterar el lado derecho $b$. En el Apéndice 1 se encuentran las respectivas fórmulas. Esta aproximación parece más pragmática que emplear análisis postoptimal, tal como aparece en Gal [3].

**b)** Luego se debe resolver el problema de programación lineal **PPL($e$)** con el método Simplex (u otro); y se debe controlar que exista solución óptima finita.

Para calcular las aristas también se puede recurrir a la teoría clásica de programación lineal presente en los textos usuales como Hadley [4], y Luenberger [5].

Las ideas geométricas parecen una vez más sugerir una manera sencilla de independizarse de las estructuras y cálculos propios de la implantación del algoritmo Simplex que se esté usando.

Simplemente conocida $x_B^*(e)$, solución óptima finita del **PPL($e$)**, se encuentran las $n$ aristas, suponiendo que no hay redundancia/escasez, resolviendo de manera consecutiva los $n$ sistemas de ecuaciones en $n$ variables que se pueden construir, alterando el valor del lado derecho, con las restricciones que se verifican en igualdad en el sistema $Ay \leq b(e), y \geq 0$. Estos sistemas se obtienen al desplazar cada vez en una cantidad fija $x_0$ sólo una determinada restricción, del total de restricciones que se verifican en igualdad en la solución óptima real $x_B^*(e)$. La solución a éste último sistema junto con $x_B^*(e)$ proporcionan los dos puntos necesarios para caracterizar a una recta, de la que sólo interesa a los efectos de **INTENT**, el vector normalizado $\delta_i$ que dirige esa arista $v_i$ en sentido interior, o sea de decrecimiento en el valor del funcional. Se debe implantar alguna clase de control de la redundancia y/o exceso del número de aristas que se puedan llegar a obtener, para conseguir el cono convexo correcto.

**c)** Definir el poliedro plano con vector $c$, que pasa por los puntos de las aristas $P_i$ es simplemente encontrar el escalar $\lambda_i$ propio de cada arista.

Si $P_i$ es el punto en la $i$-ésima arista, $\delta_i$ es el vector unitario que dirige la $i$-ésima arista $v_i$, y $x_B^*(e)$ la solución óptima finita de la "*camisa*", un punto cualquiera $x_i$ sobre la arista es:

$$x_i = x_B^*(e) + \lambda_i \, \delta_i$$

y además debe pertenecer al hiperplano $cx_i = b$.

Entonces:
$\lambda_i = \frac{b-z^*}{c\delta_i}$

Llamando $d$ el término $b - z^*$, una medida de la distancia entre el valor del funcional $z^*$ en la solución óptima finita de la camisa $x_B^*(e)$, y el hiperplano se obtiene como expresión del coeficiente:

$\lambda_i = \frac{d}{c\delta_i}$

En la práctica, para prevenir denominadores muy pequeños que pueden causar errores numéricos, se puede controlar el paralelismo del vector $c$, respecto alguna de las aristas $ar_i$, acotando el valor mínimo de $\lambda_i$.

**d)** Muestrear puntos $x$ uniformemente al azar en el poliedro, o en el politopo convexo con vértices $P_1, ..., P_n$ se puede efectuar utilizando una combinación lineal de vectores aleatorios uniformemente distribuidos en un simplex básico. Tal como aparece en Devroye [6], dado que el punto $x$ puede ser obtenido entonces como una combinación convexa de los vértices.

**e)** Muestrear puntos uniformemente al azar, en el cubo, se puede realizar empleando el método de la transformación inversa en cada variable.

**f) L**a profundidad del sondeo se puede regular mediante el valor del parámetro $\alpha$. Una vez detenido el algoritmo en la búsqueda en *descenso*, en caso de haberse obtenido alguna solución entera, se puede reiniciar la exploración ahora, en *ascenso*, hasta llegar nuevamente al vértice óptimo del **PPL($e$)**. Esto en atención a que en ciertas circunstancias puede darse el caso de obtener un punto entero subóptimo cuando se desciende. Más aún, en esta segunda fase, es aconsejable refinar el paso cuando se asciende e incluso aumentar el tamaño del muestreo en el cubo, dado que se dispone de una cota inferior del valor óptimo.

**g)** La siguiente es una organización funcional posible de **INTENT**. Se han separado a los fines de ahorrar memoria, disminuyendo el tamaño de cada programa, y a la vez simplificar la implantación, tres funciones principales :

i) definir el **PPL($e$)**,
ii) resolver el **PPL($e$)**,
iii) resolver el **PPLE** con **INTENT**.

Se utilizó en la implantación del punto ii el algoritmo Simplex presentado en el texto de Land y Powell [7].

El conjunto de programas, codificados en MSFortran 77 v5.10, permite manejar problemas de hasta 100 variables.

Los parámetros que se usan para experimentar con **INTENT** son:

espesor ($e$), paso de la búsqueda ($deltin$), tamaño del muestreo en el poliedro ($Npol$), extremo inferior del cubo ($lmin$), extremo superior del cubo ($lmax$), tamaño del muestreo en el cubo ($Ncubo$), semilla del generador de números aleatorios ($seed$).

## 4 Ilustración Numérica

Para ilustrar las ideas presentadas antes, se ha considerado un problema simple de prueba.

Se trata de resolver, usando **INTENT** el siguiente problema de programación entera:

$$\begin{array}{c} \begin{pmatrix} -1 & -2.6 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \longrightarrow \max \\ s.a. \ (\mathbf{PPLE}) \\ \begin{bmatrix} -0.2 & 2 \\ 6 & 4 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \leq \begin{pmatrix} 7.6 \\ 30 \end{pmatrix} \\ x_1 \geq 0, x_2 \geq 0, x_1, x_2 enteros \end{array}$$

Para este problema se puede observar dibujado en la Fig. 2, el convexo original cuando se relaja la condición de integridad en las variables, y el convexo asociado con el $\mathbf{PPL}(0.2)$, o sea, el que se obtiene con una camisa de espesor 0.2. En lo que sigue se escriben los vectores traspuestos.

Aquí el vector lado derecho modificado es

$b(0.2) = (8.36200047, 33.44219971)$

La solución optima real al $\mathbf{PPL}(0.2)$, es $y_B^*(0.2) = (0.0, 4.18100023)$ con un valor óptimo del funcional$zoptreal = 10.87060021$.

Las aristas unitarias son respectivamente

$ar_1 = (0.0, -1.0)$ y $ar_2 = (0.99503714, 9.95037481\,10^{-2})$

valores todos expresados en el espacio de las variables $y$.

La solución entera óptima aquí es el punto $x_B^* = (2, 4)$ y pertenece a la primera restricción.

A medida que la búsqueda se hace con el conjunto de programas que implanta **INTENT**, usando los siguientes valores en los parámetros:

$deltin = 0.5$; $Npol = 5$; $lmin = -0.2$; $lmax = 0.4$; $Ncubo = 10$; $seed = 12345$

El algoritmo **INTENT** explora sucesivamente los siguientes puntos:

$(0, 3), (0, 4), (1, 4), (1, 3), (1, 2), (2, 4), (2, 3),$

Como cabe esperar durante la ejecución de **INTENT** aparecen puntos infactibles. El punto óptimo entero $(2, 4)$ es encontrado durante el proceso de *descenso*. En el proceso *ascendente* la solución corriente actual no es mejorada, ya que el punto entero $(2, 4)$ es realmente la solución óptima de este pequeño problema.

Se puede observar, por ejemplo, que en el cuadrado $[2, 3]x[1, 2]$ todos sus puntos interiores son truncados al mismo punto entero $(2, 1)$. Esto sucede siempre e ilustra el hecho de que cada solución factible entera, en un determinado **PPLE**, tiene una determinada zona de influencia en el convexo del **PPL**$(e)$, zona donde todos sus puntos interiores son truncados al mismo punto entero. Esta observación es una buena razón para implantar **INTENT** utilizando alguna clase de método de rechazo que evite volver a verificar una y otra vez puntos enteros previamente obtenidos.

## 5 Experiencias Computacionales

En esta sección se presenta una síntesis de algunas experiencias computacionales en la resolución de pequeños problemas de programación entera. Se han diseñado esencialmente para verificar la correcta implantación de **INTENT**, y se emplearon para este fin los problemas HALDI 1-10, e IBM 1-5 según aparecen descriptos en el texto de Garfinkel y Nemhauser [8].

Las pruebas se realizaron en un microcomputador PC Pentium 166, con 16Mb de memoria, corriendo los programas como una sola tarea, en MSDOS, de Windows 95.

En todos los casos se pudo encontrar la solución óptima como se observa en el cuadro siguiente; sin embargo, establecidos valores numéricos a los parámetros de una corrida, para un cierto problema **PPLE**, se está lejos todavía de poder inferir a priori el comportamiento de **INTENT**.

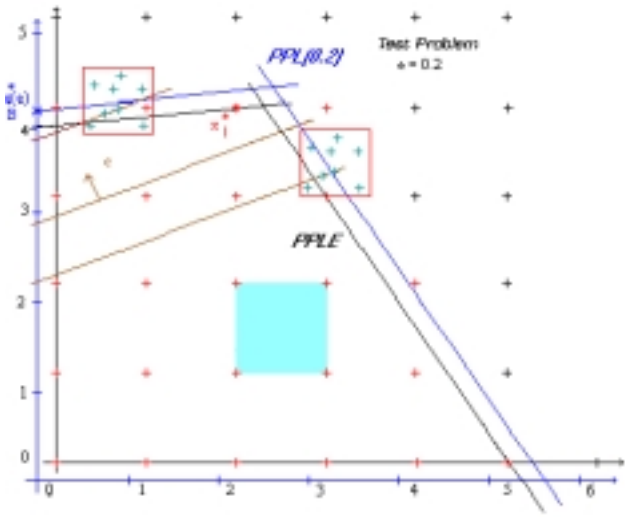


Fig. 2: Problema de prueba

La Tabla 1 presenta un resumen de algunos experimentos realizados con ese conjunto de problemas, usando la actual implantación de **INTENT**.

Ahí aparecen señalados de izquierda a derecha: la identificación del problema ($P$), el número de restricciones ($m$), el número de variables ($n$), el espesor usado en la camisa ($e$), el valor del óptimo real del **PPL($e$)** ($zoptreal$), el paso de la búsqueda ($deltin$), el tamaño del muestreo en el poliedro ($Npol$), la magnitud del extremo inferior del cubo ($lmin$), la magnitud del extremo superior del cubo ($lmax$), el tamaño del muestreo en el cubo ($Ncubo$), la semilla del generador de números aleatorios ($seed$), si consiguió o no encontrar el óptimo ($x^*_{int}$), el valor del funcional en la solución entera hallada ($z^*_{int}$), el número de puntos muestreados ($Nensy$), y el tiempo, en segundos, de CPU empleados en ese experimento ($Tcpu$).

Se puede observar que en ciertos casos algunos valores de los parámetros hacen ineficaz la búsqueda del óptimo con **INTENT**, y en otros luce ineficiente. Por ejemplo para el problema H10, se obtiene el óptimo empleando una exploración muy refinada, usando un paso ($deltin$) de 0.1, en cubos de lados $[-2.0, 1.0]$, con 50 puntos muestrea-

dos en el poliedro, y 60 puntos muestreados en cada cubo. También se lo halla con un paso de búsqueda de 5.0, utilizando en los restantes parámetros valores iguales. La diferencia de esfuerzo computacional entre ambas resoluciones se ve reflejada en la relación 208 veces en tiempo y 34 veces en la cantidad de puntos muestreados. Este hecho ilustra, nuestro desconocimiento a priori, de cuales valores numéricos en los parámetros son adecuados para hallar, de manera eficiente, el óptimo entero en un determinado problema.

En ciertos problemas se logran encontrar soluciones múltiples, ya que en la implantación realizada se verifica si el nuevo punto candidato al óptimo es distinto del actual, y se guardan cada vez los sucesivos mejores valores obtenidos. Por otro lado en la implantación actual no se controla la integridad de la solución óptima real $y_B^*(0.0)$, o sea con camisa de espesor 0.

A título informativo, el mismo experimento con el problema Haldi 10 (H10), que tarda 1718.18 seg. en el equipo Pentium, ejecutándose en condiciones similares en un equipo 486 con 24MB de memoria consume 8105.91 seg. para obtener el óptimo; un factor de 4.7 veces más.

En otro orden de cosas, el problema Haldi 2 (H2), ejecutándose como una tarea de Windows 95 tarda 31.86 seg., si se ejecuta el mismo programa **INTENT** en la máquina Pentium, pero en modalidad sólo MSDOS tarda 6.15 seg. Un 19.3% del tiempo original.

| P | m | n | e | zoptreal | deltin | Npol | lmin | lmax | Ncubo | Seed | X*int | Z*int | ensy | Tcpu |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| H1 | 6 | 5 | 0.1 | 9.31 | 2.0 | 30 | -1.5 | 1.0 | 50 | 1234567 | si | 7 | 18000 | 4.01 |
| H2 | 6 | 5 | 0.5 | 12.34 | 0.2 | 300 | 0.0 | 0.0 | 0 | 3141567 | si | 8 | 47700 | 31.86 |
| H3 | 6 | 5 | 2.0 | 22.99 | 0.3 | 400 | 0.0 | 0.0 | 0 | 98712345 | si | 10 | 92800 | 45.81 |
| H4 | 6 | 5 | 0.7 | 11.75 | 0.6 | 50 | -1.3 | 1.3 | 60 | 2560859 | si | 7 | 162000 | 16.13 |
| H5 | 6 | 5 | 0.4 | 107.02 | 0.5 | 80 | -1.4 | 1.3 | 100 | 54321 | si | 76 | 4080000 | 1331.88 |
| H6 | 5 | 5 | 0.5 | 142.42 | 1.0 | 75 | -1.0 | 1.0 | 40 | 11223377 | si | 106 | 981000 | 274.25 |
| H7a | 4 | 5 | 0.0 | 110.0 | 1.0 | 50 | 0.0 | 0.0 | 0 | 123 | si | 110 | 8700 | 29.99 |
| H8a | 4 | 5 | 0.0 | 140.0 | 1.4 | 75 | -1.0 | 1.0 | 50 | 987651 | si | 140 | 566250 | 138.80 |
| H9 | 9 | 6 | 0.5 | 16.31 | 0.4 | 45 | -1.1 | 1.2 | 60 | 45456677 | si | 9 | 318600 | 32.84 |
| H10 | 16 | 12 | 0.1 | 20.51 | 5.0 | 50 | -2.0 | 1.0 | 60 | 98712333 | si | 17 | 36000 | 8.24 |
| H10 | 16 | 12 | 0.1 | 20.51 | 2.5 | 50 | -2.0 | 1.0 | 60 | 98712333 | si | 17 | 54000 | 12.34 |
| H10 | 16 | 12 | 0.1 | 20.51 | 1.0 | 50 | -2.0 | 1.0 | 60 | 98712333 | si | 17 | 135000 | 41.03 |
| H10 | 16 | 12 | 0.1 | 20.51 | 0.1 | 50 | -2.0 | 1.0 | 60 | 98712333 | si | 17 | 1242000 | 1718.18 |
| H10 | 16 | 12 | 0.3 | 24.13 | 5.0 | 50 | -2.0 | 1.0 | 60 | 98712333 | no | - | 15000 | 3.46 |
| H10 | 16 | 12 | 0.3 | 24.13 | 2.5 | 50 | -2.0 | 1.0 | 60 | 98712333 | si | 17 | 81000 | 22.24 |
| H10 | 16 | 12 | 0.3 | 24.13 | 1.5 | 50 | -2.0 | 1.0 | 60 | 98712333 | si | 17 | 117000 | 33.12 |
| IBM1 | 7 | 7 | 0.3 | -6.92 | 0.5 | 25 | -1.0 | 1.0 | 25 | 12345 | si | -8 | 9375 | 7.42 |
| IBM2 | 7 | 7 | 0.5 | -4.37 | 0.6 | 100 | 0.0 | 0.0 | 0 | 12332111 | si | -7 | 900 | 2.92 |
| IBM3 | 10 | 7 | 0.0 | -179.77 | 7.0 | 150 | 0.0 | 0.0 | 0 | 654321 | si | -187 | 5400 | 12.08 |
| IBM4 | 15 | 15 | 1.0 | -3.82 | 0.6 | 20 | -1.0 | 1.0 | 40 | 123 | si | -11 | 17600 | 15.21 |
| IBM5 | 15 | 15 | 1.5 | -4.24 | 1.3 | 20 | -1.2 | 1.4 | 35 | 123 | si | -14 | 8400 | 8.19 |

Tabla 1

La observación de una salida típica, ver apéndice 2, para la resolución del problema Haldi 10 permite tomar idea de ciertos aspectos del comportamiento de **INTENT**, durante el proceso de encontrar la solución óptima. La salida presentada se corresponde con la fila 10 de la tabla anterior.

Los parámetros de **INTENT** tienen en este caso los valores:

$ESPESOR\,DE\,LA\,CAMISA 1.000000E-01$
$DELTIN = 5.0$
$NMUESTRA\,EN\,POLIEDRO\,ES = 50$
$LADO\,INFERIOR\,DEL\,CUBO = -2.0$
$LADO\,SUPERIOR\,DEL\,CUBO = 1.0$
$TAMANIO\,DE\,LA\,MUESTRA\,EN\,EL\,CUBO = 60$
$SEMILLA = 98712333$

Durante el proceso de descenso, partiendo de una solución infactible, $-100$ en todas sus componentes, con un valor de funcional asociado $-\infty$, se consiguen sucesivamente varios puntos factibles, durante el proceso de *descenso* ($D$), y *ascenso* ($A$) desde $x_B^*(0.1)$. Los sucesivos valores obtenidos aparecen en la Tabla 2.

El vector siguiente muestra el valor de la solución óptima finita del **PPL**$(0.1)$ camisa

$$x_B^*(0.1) = (-.10, -.10, .03, .80, -.10, .81, -.10, -.10, 1.62, 9.04, .00, 10.06)$$

| Dirección | Solución | funcional |
|---|---|---|
| D | ( 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0) | 11 |
| D | ( 0, 0, 0, 1, 0, 1, 0, 0, 0, 8, 0, 7) | 15 |
| D | ( 0, 0, 1, 1, 0, 1, 0, 0, 0, 7, 0, 8) | 15 |
| A | ( 0, 0, 0, 1, 0, 1, 0, 0, 0, 7, 0, 8) | 15 |
| A | ( 0, 0, 0, 1, 0, 1, 0, 0, 0, 7, 0, 9) | 16 |
| A | ( 0, 0, 0, 1, 0, 1, 0, 0, 0, 8, 0, 9) | 17 |

Tabla 2

## 6 Conclusiones

Se han introducido algunas ideas que permiten especificar un nuevo algoritmo heurístico para resolver problema de programación lineal entera. Los primeros ensayos computacionales con algunos problemas pequeños, han dado resultados promisorios. Se requiere hacer mucha más experimentación para tratar de inferir que valores son efectivos como parámetros del algoritmo **INTENT**. No se ha realizado un análisis computacional, dado que la intención primordial era explorar la viabilidad del conjunto de ideas presentadas. Los tiempos de ejecución están altamente influídos por el tamaño de los respectivos muestreos, y la magnitud del paso de búsqueda. Como era de esperar, el proceso ascendente es computacionalmente caro, cuando simultáneamente se refina el paso y se incrementa el tamaño en los muestreos. La dificultad principal actual del algoritmo **INTENT** es la inexistencia de un criterio de optimalidad más

eficiente que el descripto aquí para detener la exploración.

## References

[1] R. A. PASTORIZA, "**INTENT** un algoritmo interior para la resolución de problemas de programación entera", Papel de trabajo, UCV, Facultad de Ciencias, Escuela de Computación, Laboratorio de Modelos Matemáticos, octubre de 1996.

[2] R. A. PASTORIZA, "**INTENT** Una heurística para programación entera", RT 98-02, UCV, Facultad de Ciencias, Escuela de Computación, Laboratorio de Modelos Matemáticos, marzo de 1998.

[3] T. GAL, *Postoptimal Analyses, Parametric Programming, and related Topics*, ed. McGraw-Hill International Book Company, New York,1979.

[4] G. HADLEY, *Linear Programming*, ed. Addison-Wesley Publishing Co, Reading, Massachusetts, 1962.

[5] D. G. LUENBERGER, *Linear and Nonlinear Programming*, ed. Addison-Wesley Publishing Co, Reading, Massachusetts, 1973.

[6] L. DEVROYE, *Non-Uniform Random Variate Generation*, ed. Springer-Verlag, New York, 1986.

[7] A. H. LAND y S. POWELL, *Fortran Codes for Mathematical Programming: Linear, Quadratic and Discrete*, ed. J. Wiley & Sons, London, 1979.

[8] R. S. GARFINKEL y G. L. NEMHAUSER, *Integer Programming*, ed. J. Wiley, New York, 1972.

**Apéndice 1**

Dado el problema de programación entera

$$\begin{array}{c} Max \quad cx \\ s.a. \\ Ax \leq b \ (\mathbf{PPLE}) \\ x \geq 0 \quad x \quad entero \end{array}$$

se trata de construir un problema de programación lineal **PPL**$(e)$ cuyo convexo contenga al convexo definido por el sistema lineal de desigualdades $Ax \leq b$, $x \geq 0$, dado por el **PPLE**, a una distancia positiva fija $e$. La matriz $A$ con dimensiones $mxn$.

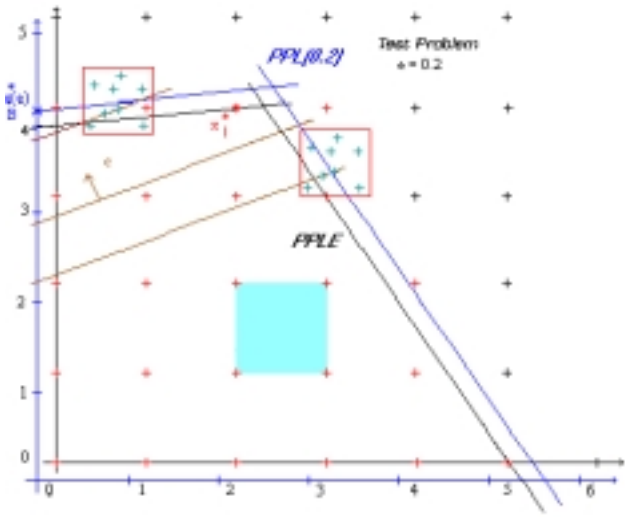


Fig. 3: Hiperplano paralelo

Para calcular un hiperplano paralelo a un hiperplano $\alpha$ cualquiera dado por alguna de las $m$ restricciones del **PPLE**, la figura siguiente sirve de guía.

Se tiene que :

$x = P + \varepsilon \quad con \quad \varepsilon = e\, an_\alpha$

donde $an_\alpha$ es el vector normalizado $a_\alpha$ que dirige el hiperplano $\alpha$, y $e$ es un escalar positivo, el valor de espesor especificado.

Como se debe cumplir que

$a_\alpha x = b_\alpha(e)$

se puede ver fácilmente que

$b_\alpha(e) = b_\alpha + e\, an_\alpha$

Entonces se puede definir el vector $an$ como aquel vector de $m$ componentes donde el $i$-ésimo componente es la norma del $i$-ésimo vector $a_i$, $i$-ésima fila de la matriz $A$.

$an = (\parallel a_i \parallel_i) \quad con \quad i = 1, 2, ..., m$

Simultáneamente se deben también cambiar las variables, esto conduce a :

$y = x + e\, 1$

donde 1 es un vector con (1) unos en cada una de sus componentes.

La sustitución en las desigualdades proporciona la expresión:

$Ax = Ay - e\, A1$

llamando $sa$ al vector cuyas componentes son las sumas de los valores de las filas de la matriz $A$, se tiene :

$sa = (sa_i) \quad con \quad sa_i = \sum_j a_{ij}, \quad i = 1, 2, ..., m$

De manera similar se puede definir $sc$, un escalar que represente la suma de todos los componentes del vector $c$; quedando entonces el funcional expresado en términos de la nueva variable $y$ como

$Max \quad cy - e\, sc$

Como en realidad el valor de la constante $sc$ no interesa a los fines de **INTENT**, se la puede descartar en la formulación del **PPL($e$)**.

En resumen el problema de programación lineal que se busca es :

$$\begin{array}{c} Max \quad cy \\ s.a. \\ Ay \leq b + e\, sa + e\, an \ (\mathbf{PPL}(e)) \\ y \geq 0 \end{array}$$

La única precaución adicional que se debe tener con la solución óptima $y_B^*$ del **PPL(e)** es la necesidad de volver a las variables originales $x$, para calcular el vértice $x_B^*(e)$ y las aristas $ar_i$ en términos de las variables del problema original.

En la implantación se calcula la expresión antes indicada del lado derecho $b(e) = b + e\,sa + e\,an$, conociendo la matriz $A$, el lado derecho $b$ del problema original **PPLE**, y el valor del espesor $e$.

**Apéndice 2**

Se presenta una salida típica de un problema de prueba, esta corresponde al problema H10.

```
ESPESOR DE LA CAMISA 1.000000E-01
OPTIMO REAL ===> 20.517600
SOLUCION OPTIMA REAL
-.10 -.10 .03 .80 -.10 .81
-.10 -.10 1.62 9.04 .00 10.06
DIRECCION INTERIOR DESDE EL OPTIMO REAL
1 2 3 4 5 6
7 8 9 10 11 12
...46 .83 .36 .21 .05 .52
...84 .83 -1.92 -3.24 -.12 -2.71
DELTIN = 5.000000
NMUESTRA EN POLIEDRO ES = 50
LADO INFERIOR DEL CUBO = -2.000000
LADO SUPERIOR DEL CUBO = 1.000000
TAMANIO DE LA MUESTRA EN EL CUBO = 60
SEMILLA = 98712333
****************************
>>> VALORES INICIALES DE INTENT <<<
LA SOLUCION ES ..... >
-100 -100 -100 -100 -100 -100
-100 -100 -100 -100 -100 -100
.....
BAJANDO
FUNCIONAL = 11.000000
LA SOLUCION ES ..... >
0 1 0 0 0 0
0 11 0 0 0 0
FUNCIONAL = 15.000000
LA SOLUCION ES ..... >
0 0 0 1 0 1
0 0 0 8 0 7
FUNCIONAL = 15.000000
LA SOLUCION ES ..... >
0 0 1 1 0 1
0 0 0 7 0 8
```

```
++++++++++++++++++++++++++++
SUBIENDO
FUNCIONAL = 15.000000
LA SOLUCION ES ..... >
0 0 0 1 0 1
0 0 0 7 0 8
FUNCIONAL = 16.000000
LA SOLUCION ES ..... >
0 0 0 1 0 1
0 0 0 7 0 9
FUNCIONAL = 17.000000
LA SOLUCION ES ..... >
0 0 0 1 0 1
0 0 0 8 0 9
****************************************
ENSAYOS 36000 TIEMPO = 9.279999999998836 SEGUNDOS
****************************************
```