Tesis de Licenciatura en Ciencias de la Computación

"Optimización de la Paginación de Avisos Clasificados"

Abramzón, Federico – L.U. 560/90
fede@dc.uba.ar

Maceratesi, Martín – L.U. 47/90
martinm@dc.uba.ar

Quevedo, Oscar – L.U. 351/90
quevedo_oscar@yahoo.com

Directora
Lic. Irene Loiseau

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Argentina

Diciembre 2001

Agradecimientos

Queremos expresar nuestro agradecimiento a aquellas personas que, de una u otra forma, nos ayudaron a concretar este trabajo de tesis.

A nuestra directora, Lic. Irene Loiseau, por su dedicación, paciencia y constante estímulo.

Al Lic. Fernando Urtubey y la Lic. Andrea Efron, por la revisión y los comentarios de las últimas versiones de nuestro trabajo.

Federico Abramzón, Martín Maceratesi y Oscar Quevedo

A mi esposa Natalia, por su apoyo, comprensión y permanente estímulo durante todo este ultimo tiempo. A mis padres Marta y Rogelio, por brindarme la posibilidad de realizar mis estudios en Buenos Aires. A mi hermano Diego, por estar siempre. Y obviamente, a mis compañeros de tesis: Martín y Oscar, por la cordialidad y compromiso asumido. Dedicado a mis padres.

Federico Abramzón

A Fede y Oscar, por creer en este proyecto y comprometerse de lleno en él. Sin ellos, no hubiera sido posible concretarlo. A mi familia, por la contención, el apoyo incondicional, y el continuo empuje y estímulo. A mi padrino Eduardo, quien una vez me planteó este problema, y con quien di mis primeros pasos, en este tema. A mis amigos, por alentarme constantemente. Dedicado a mi hijito, Nicolás.

Martín Maceratesi

A mis padres, Norma y Francisco, quienes han hecho un gran esfuerzo para que pudiera realizar mis estudios. A Laura, por su comprensión y paciencia, y quien con su apoyo e insistencia, me ha incentivado a completarlos. A todos aquellos que en todo este tiempo me han alentado y apoyado. Y por supuesto, a Fede y Martín, ya que sin ellos, esto no hubiera sido posible.

Oscar Quevedo

Resumen

En este trabajo abordamos el **Problema de la Paginación de Avisos Clasificados**, que consiste en distribuir los avisos y el texto, a lo largo de las páginas de una publicación (ya sea, los rubros clasificados de un diario o un directorio telefónico – del tipo de *Las Páginas Amarillas*).

La distribución "compacta" y "armónica" de los avisos y el texto, es una tarea fundamental y costosa en la producción de este tipo de publicaciones comerciales. En este sentido, no sólo debe tenerse en cuenta la minimización de espacio (y desperdicio asociado), sino también, deben considerarse una serie de criterios gráficos. Estos últimos definen, sobre todo, la manera de posicionar los avisos en las páginas de estas publicaciones.

Usando como punto de partida la heurística inicial presentada por R. Johari, J. Marks, A. Partovi y S. Shieber, se diseñó e implementó un algoritmo original de *Búsqueda Tabú*, como guía por sobre una heurística de mejora adaptada, con el objeto de "escapar" de óptimos locales.

El prototipo implementado resultó ser efectivo, tanto para casos generados aleatoriamente, como para casos tomados de publicaciones reales. Además, desarrollamos una interfase gráfica amigable para facilitar la interacción con el usuario.

Este trabajo es complementado con un amplio relevamiento sobre productos comerciales disponibles en el mercado.

Abstract

In this work we deal with the **Pagination and Layout of Classified Ads Problem**, which consists in allocating displad ads and text along the pages of a publication (it could be a classified ads of a newspaper, or a commercial telephone directory, like *Yellow Pages*).

The generation of a "compact" and "harmonious" layout for the publication is the main and the most expensive task of the process. However, reducing space (and its associated waste) is not the only issue to take into account. A variety of graphics criteria must also be considered. These criteria define mainly the way ads are laid out on the pages.

Based on an initial heuristic introduced by R. Johari, J. Marks, A. Partovi y S. Shieber, we have designed and implemented an original *Tabu Search* algorithm, as a guide over an adapted improvement heuristic, with the purpose of avoiding local optimums.

The implemented prototype was effective for both random generated test cases and test cases drawn from a real yellow-pages publication. We also developed a friendly graphic interface to improve the interaction with the user.

This work has been completed with a wide survey of commercial products available on the market.

INDICE

Figuras

Tablas

1 INTRODUCCION

El **Problema de la Paginación de Avisos Clasificados**, que consiste en distribuir los avisos clasificados a lo largo de una publicación, de una manera "compacta" y "armoniosa", es aún en la actualidad una de las tareas de mayor costo (en términos de tiempo y recursos) en el proceso de la producción de estas publicaciones comerciales. Esta clase de problema aparece tanto en los rubros clasificados de un diario, como así también en los directorios telefónicos del tipo de *Las Páginas Amarillas* o *Páginas Doradas*.

En especial, cuando una publicación posee una gran cantidad de avisos, la posibilidad de automatizar este proceso conlleva un importante ahorro de trabajo y una considerable mejora de la calidad del producto.

Uno de los objetivos planteados, el de que tan "compacta" resulta la publicación, impacta en forma directa en el costo de impresión de la misma. En este sentido, dado que la superficie de avisos (*destacados* y *lineales*) está fijada, el ahorro estará dado, exclusivamente, por la minimización del material de relleno. Por lo tanto, la generación de publicaciones más "compactas", sin desmedro de la calidad en cuanto a la composición de los avisos en sus páginas, será fundamental.

La "armoniosa" composición (layout) de los avisos clasificados en las páginas de la publicación es el resultado de una serie de criterios y convenciones que tienen que ver, por sobre todo, con ciertas restricciones en el formato de las páginas. Estas convenciones varían, en general, de publicación en publicación, y de país en país. En nuestro trabajo, tomaremos como referencia el tipo de layout que se utiliza en *Las Páginas Amarillas*. En estas publicaciones, los *avisos destacados* y los *lineales* se disponen en páginas de cinco columnas, ubicándose los *lineales* por encima de los *destacados*. Los *avisos destacados* de un rubro dado deberían estar ubicados en las páginas de dicho rubro, es decir, en la página de inicio o fin de la sección en cuestión (o en una página entre ellas). Al mismo tiempo, para una sección dada, los *destacados* más pequeños no pueden aparecer en una página anterior a la de un *destacado* de mayor superficie. Por último, en el caso de los *avisos lineales*, hay que evitar que se produzca un "corte de columna" inmediatamente después de un *encabezado de sección*.

Si bien los criterios adoptados en nuestro trabajo toman como referencia los utilizados en las publicaciones de *Las Páginas Amarillas*, el enfoque aquí propuesto puede ser fácilmente adaptado para tener en cuenta otras consideraciones.

1.1 Trabajos previos

El **Problema de Paginación** fue tratado en numerosos trabajos previos. En este sentido, del profundo relevamiento bibliográfico (y vía web) que realizamos, pudimos recabar diferentes enfoques utilizados, tanto para la problemática planteada, como también para una amplia gama de variantes relacionadas. Encontramos al mismo tiempo, numerosos productos comerciales, también orientados a las diferentes problemáticas de esta clase de publicaciones.

Muchas de las publicaciones científicas y académicas analizadas, y de los productos comerciales encontrados, están más orientados a otros problemas relacionados con el que tratamos aquí:

- Armado de documentos: ya sea para su publicación impresa o para su distribución digital (CD-Rom, DVD o Web).
- Formateo de texto: desde el punto de vista de la justificación, interletrado, interlineado, etc.
- Automatización del armado de diarios y revistas: pero más orientado al estilo utilizado en las secciones de noticias, que a rubros clasificados.
- Optimización de métodos de impresión: manejo de pliegos, separación de colores, entre otras cosas.

En el estudio aquí realizado separamos por un lado los trabajos teóricos relacionados, y por otro, algunos productos comerciales encontrados.

1.1.1 Publicaciones científicas y académicas

Desde el punto de vista teórico, encontramos diferentes enfoques aplicados a esta variedad (antes mencionada) de problemáticas relacionadas al **Problema de Paginación** ([P81], [IIDYFT], [RJHMES], [CHLKOT], [WW94], [HMWB], [GNG96], [JMPS97], [K95], [A94], [VALNY], [YA90], [A89] y [AV89]). En este sentido, estos problemas son encarados mediante diversas técnicas: *Métodos Basados en Reglas*, *Métodos Basados en Restricciones*, *Programación Dinámica* y *Búsqueda Heurística*.

Los *Métodos Basados en Reglas*, aparentemente, han sido aplicados con éxito desigual tanto al **Problema de Paginación** como así también a algunas de sus variantes. Si bien la clase de reglas "if-then" pueden resultar una buena forma de definir una *función objetivo*, parecerían en primera instancia no ser del todo adecuadas para representar completamente esta clase de *Problemas de Optimización Combinatoria*. De hecho, su aplicación práctica a otros problemas de estas características, como por ejemplo el "Problema de la Ubicación de Etiquetas en Mapas Cartográficos", resultó desplazada por otras técnicas, como ser la *Búsqueda Heurística*.

Por su parte, algunos trabajos describen soluciones al **Problema de Paginación** mediante *Métodos Basados en Restricciones* [GNG96]. Algunos sistemas basados en este enfoque son descriptos específicamente más adelante.

Ciertas variantes restringidas del **Problema de Paginación**, en especial aquellas en las que los avisos deben mantener un determinado *orden* (por ejemplo, alfabético), son factibles de implementar mediante técnicas de *Programación Dinámica* (ver [P81]). Este enfoque presenta, principalmente, dos desventajas. Por un lado, a pesar de que esta clase de algoritmos son *polinomiales*, en la práctica, puede tornarse prohibitiva su aplicación debido al tamaño del problema. Por otro lado, este tipo de implementaciones generalmente resultan ser muy específicas y particulares de la variante de problema considerada.

A continuación, describimos sintéticamente algunos de los trabajos estudiados:

"Automated pagination of the generalized newspaper using Simulated Annealing" [K95]

Este trabajo contribuye en la mejora del sistema comercial VIP (ver Apéndice D). Este sistema utilizaba un algoritmo de *Búsqueda Heurística* en grafos. Mediante este trabajo se ha mejorado el algoritmo aplicando *Simulated Annealing*. También se analizan y comparan dos implementaciones experimentales usando esta misma técnica a la composición de páginas individuales.

"Embedded optimization tasks in a Pagination Problem" [A94]

Como la complejidad del problema hace difícil cualquier formulación matemática, este trabajo define sub-problemas especiales que pueden ser planteados de una manera exacta. Concluye que este tipo de formulación ayuda a encontrar mejores algoritmos y a construir implementaciones de sistemas de paginación más flexibles.

"Automatic Page Layout for Electronic News Distribution" [VALNY]

Este trabajo presenta una extensión a la paginación de avisos para la composición de páginas editoriales, configurables para aplicaciones que van desde distribución de noticias electrónicas para ser desplegadas en un monitor, hasta páginas completas de diarios. El principal criterio para la composición, además de una estética aceptable usando artículos de distintos tamaños y formas, es la minimización de espacios vacíos.

"Automatic Yellow-Pages Pagination and Layout" [JMPS97]

Este trabajo formula una versión del "Problema de Paginación y Layout de Páginas Amarillas" como un *problema de optimización combinatoria*. Las tareas de posicionamiento de *avisos destacados* y texto se realizan minimizando la cantidad de páginas y maximizando el estilo de layout, sujetas a las restricciones impuestas por los requerimientos del formato de página, y relaciones entre los avisos y el texto.

El algoritmo propuesto se basa en la técnica de *Búsqueda Heurística* de *Simulated Annealing* [KGV83]. La principal contribución que realiza este trabajo es la representación de una *solución candidata* y el conjunto de *operadores de movimiento*. Como conclusión, sostiene que el algoritmo propuesto obtiene soluciones significativamente mejores, aplicado a una guía telefónica real.

"Knowledge-based newspaper pagination" [YA90]

Encara el "Problema de la Distribución de Avisos" desde el punto de vista de *Sistemas Basados en Conocimiento*. Esta clase de problemas pueden ser vistos como sub-tareas, las cuales pueden ser resueltas utilizando soluciones predefinidas para problemas conocidos. Las problemáticas específicas de la publicación (características del diario, avisos, proceso de producción, etc.) son las que definen las restricciones a tenerse en cuenta en el proceso.

"Knowledge-Based Integration of Newspaper Design and Layout" [A89]

Este trabajo formula el "Problema de Diseño y Composición de Diarios" como un *Problema Basado en Conocimiento*. Presenta sistemas que utilizan este esquema para resolver sub-tareas del proceso, tales como la paginación y la composición de avisos clasificados.

"Knowledge-based Pagination" [AV89]

Presenta un sistema para resolver el "Problema de la Paginación de Páginas Amarillas". Analiza las características especiales de este tipo de directorios. Muestra como las heurísticas de la persona encargada del diseño pueden ser imitada mediante un *Sistema Basado en Conocimiento*. Finalmente, discute sus posibles extensiones a otras tareas de la paginación.

1.1.2 Productos comerciales

En el mercado existen numerosos productos que intentan resolver de una u otra forma el **Problema de Paginación** de publicaciones. A continuación enumeramos las principales características que poseen:

- Integrados a suite de productos que administran la información de una Editorial.
- Plataformas abiertas.
- Visualizador de las páginas de una publicación: búsqueda, navegación, zoom, composición, identificación.
- Interacción con el usuario para la alteración de características (posición, propiedades, etc.) de los elementos de una publicación.
- Generación de las páginas en formato PostScript o EPS para impresión.
- Generación de las páginas para ser importadas a aplicaciones de diseño para su retoque gráfico.
- Generación automática de anclas (elementos *lineales* que hacen referencia a *avisos destacados*).
- Parametrización del estilo del layout de páginas.
- Justificación vertical de los *avisos lineales*.
- Generación automática de *avisos de relleno* para publicidad.

En el Apéndice D, profundizamos sobre algunas herramientas comerciales disponibles actualmente en el mercado. Si bien, la información técnica disponible de estos productos es muy reducida (principalmente se concentran en características funcionales), varias de estas, dicen utilizar "mecanismos avanzados de la Inteligencia Artificial" para el proceso de paginación.

Como tarea preliminar al desarrollo de este trabajo, analizamos en profundidad el sistema *Calligramme Directory* que está implementado en la división *Las Páginas Amarillas*, del Grupo Telecom. En este caso, tuvimos la oportunidad de ver el producto comercial en funcionamiento.

1.2 Enfoque utilizado

El **Problema de Paginación**, en los términos planteados, tiene una dificultad intrínseca. Podemos pensarlo como la combinación de dos variantes del problema de Bin-Packing. Por un lado, la distribución de los *avisos destacados* en las sucesivas páginas de la publicación. Y por el otro, los "cortes" de *avisos lineales* a ubicar en el espacio remanente. Si bien existen algoritmos de programación dinámica ([KP81]) que resuelven de manera óptima el problema de "Cortes de Líneas" en tiempo lineal, la dificultad radica en que los "cortes" de *avisos lineales* están supeditados a la configuración que toman los *avisos destacados* en la página; y a su vez, la ubicación de estos últimos depende de la posición de los *lineales*, y en particular, de los encabezados de sección. Esto hace que, en este caso, ambos problemas deban resolverse simultáneamente.

Por otra parte, la gran variedad de restricciones dificulta considerablemente la formulación del **Problema de Paginación** mediante un modelo matemático de optimización.

El **Problema de Paginación** pertenece a la clase **NP-hard** (ver [JMPS97]), por lo que no se conoce algoritmo polinomial exacto para resolverlo. Por lo tanto, la búsqueda de "la mejor solución" se torna computacionalmente intratable mediante la utilización de algoritmos exactos.

Si la entrada del algoritmo consistiera en unos pocos avisos a ubicar, bastaría con probar "todas" las posibles distribuciones (intercambiando las posiciones de los distintos avisos - si se consideran sólo los *destacados*), y conservar la "mejor" solución. Pero en la práctica, lamentablemente la entrada está formada por una cantidad considerable de avisos clasificados, y el número de combinaciones crece vertiginosamente. Un algoritmo exacto sería, en este caso, de orden factorial respecto del tamaño de la entrada (O(n!)). Esto es impensable si se considera que existen publicaciones de frecuencia diaria.

Por tratarse de un problema de optimización combinatoria, podríamos abordarlo desde diversas perspectivas. Entre las más comunes, podemos mencionar: el uso de técnicas *Branch & Cut*, y el empleo de *heurísticas de tiempo polinomial determinístico*.

En nuestro trabajo decidimos recurrir a ***Técnicas Heurísticas*** para su solución (o aproximación a la misma). De esta forma, pueden obtenerse "buenas soluciones" (aunque no necesariamente óptimas) del problema a tratar. En este sentido, partiremos de una *solución inicial* (no necesariamente factible), generada mediante heurísticas constructivas, a la que aplicaremos heurísticas de mejora para la obtención de "buenas" soluciones. En esta tesis, utilizamos una versión adaptada por nosotros de la metaheurística de *Búsqueda Tabú*, como guía de las heurísticas de mejora implementadas.

1.3 Organización del trabajo

Comenzamos describiendo el problema y sus variantes con mayor profundidad en el Capítulo 2. Luego, en el Capítulo 3, realizamos un desarrollo teórico sobre el mecanismo de *Búsqueda Tabú* con el propósito

de sentar las bases del trabajo. Desarrollamos el algoritmo propuesto en el Capítulo 4, y comentamos los detalles de su implementación en el Capítulo 5. En el Capítulo 6, presentamos las pruebas efectuadas sobre el prototipo desarrollado, y en el Capítulo 7 concluimos con la discusión de las características y limitaciones de nuestra propuesta.

2 DESCRIPCION DEL PROBLEMA

En el presente capítulo enunciamos, con mayor detalle, la problemática a tratar. Introducimos, en este sentido, la terminología utilizada en el resto del trabajo. A partir de allí, presentamos una serie de criterios y convenciones, específicas del problema aquí planteado. Describimos, en forma preliminar, los objetivos a optimizar. Y finalmente, mencionamos otras consideraciones a tener en cuenta.

Como ya se mencionó en la Introducción, el problema a tratar consiste en encontrar una manera adecuada de disponer los avisos clasificados de una publicación, de acuerdo a una serie de criterios gráficos. En este sentido, existen numerosas reglas y consideraciones referidas, en su mayoría, a la ubicación que pueden tomar los distintos tipos de avisos en las páginas de estas publicaciones. Todos estos criterios deben considerarse en la búsqueda de la solución. Esta clase de problema es aplicable tanto a los rubros clasificados de un diario, como así también a directorios telefónicos del tipo de *Las Páginas Amarillas* o *Páginas Doradas*.

El **Problema de la Paginación de Avisos Clasificados** consiste entonces en distribuir los avisos clasificados a lo largo de una publicación, de una manera "compacta" y "armoniosa". El qué tan "compacta" resulte la publicación, impactará en forma directa en el costo de impresión de la misma. Ahora bien, dado que la superficie de avisos intervinientes en el proceso de paginación está fijada, el ahorro estará dado exclusivamente, por la minimización del material de relleno. Por lo tanto, la generación de publicaciones más "compactas", sin desmedro de la calidad en cuanto a la composición de los avisos en sus páginas, será fundamental. Por su parte, la "armoniosa" composición (layout) de los avisos clasificados en las páginas de la publicación es el resultado de una serie de criterios y convenciones que tienen que ver, por sobre todo, con ciertas restricciones en el formato de las páginas. En general, estas convenciones varían de publicación en publicación, y de país en país.

Debido a que una publicación comercial de este tipo posee una gran cantidad de avisos, la posibilidad de automatizar el proceso de la paginación conlleva un importante ahorro (fundamentalmente, en términos de tiempo y recursos), y una considerable mejora de la calidad del producto. En el caso particular de los avisos clasificados de un diario, este proceso se realiza generalmente al "cierre de edición". En este sentido, la posibilidad automatizar y acelerar el proceso de la paginación permitiría extender dicho cierre, de manera de incluir una mayor cantidad de avisos en la publicación, incrementando los ingresos de allí derivados. Tendrá entonces que fijarse cierto "compromiso" entre cuán buena es la solución obtenida y el tiempo invertido para alcanzarla.

2.1 Elementos que componen una publicación

Como una definición previa, vamos a considerar que cada página de la publicación está formada por una cantidad fija de filas y columnas. Sus dimensiones suelen variar de publicación en publicación.

Los avisos a distribuir en las páginas están organizados en *rubros* (o *secciones*). Cada *sección* posee un *encabezado* que debe colocarse delante de todos los avisos asociados. Los *encabezados de sección* tienen en general, dependiendo de la publicación considerada, una columna de ancho, y una o varias filas de alto.

Existen distintos tipos de avisos a ubicar en las páginas. Los *avisos destacados* son rectángulos de diferentes dimensiones conteniendo texto artístico e imágenes. Estos pueden abarcar más de una columna de ancho y, en general, se disponen de manera tal de formar "bloques compactos" de avisos. Los *avisos lineales* consisten en una o más líneas de texto, de una columna de ancho. Estos están, generalmente, ordenados alfabéticamente dentro de cada *sección*. En ciertos casos, existen *avisos lineales* especiales

denominados *anclas*, cuya función es la de referenciar *avisos destacados* relacionados, indicando su ubicación dentro de la publicación.

Los espacios remanentes, una vez ubicados los *avisos destacados* y los *lineales*, son llenados con material de *relleno*. Esto puede considerarse, en primera instancia, como espacio desperdiciado en la publicación, y en la práctica suele utilizarse para publicidad propia y/o de terceros. Eventualmente, también pueden forzarse intencionalmente, generando áreas reservadas para fines publicitarios. En este sentido, podemos distinguir a su vez tres clases de *avisos de relleno*:

> *Fillers*: son espacios que pueden generarse al final de las columnas de *avisos lineales*, por evitar que se produzcan cortes "indeseados" (por ejemplo, inmediatamente después de un *encabezado de sección*).
>
> *Burbujas*: son espacios "atrapados" entre los *avisos destacados*.
>
> *Espacios reservados (o fillers "fijos")*: son las áreas reservadas que, como mencionamos anteriormente, se fuerzan con fines publicitarios.

Como veremos más adelante, el hecho de distinguir entre *burbujas* y *fillers*, nos permitirá darle diferente tratamiento al desperdicio.

Pueden existir además, dependiendo del tipo de publicación, *líneas de continuación*, para indicar que el *rubro* en cuestión "continúa en otra página" o "viene de otra página".

En nuestro trabajo, tomaremos como referencia el tipo de layout que se utiliza en los directorios telefónicos de *Las Páginas Amarillas*. A continuación, en la Figura 1, mostramos a modo de ejemplo una página "tipo" de una publicación comercial. Indicamos allí los diferentes elementos que componen una publicación de esta clase.

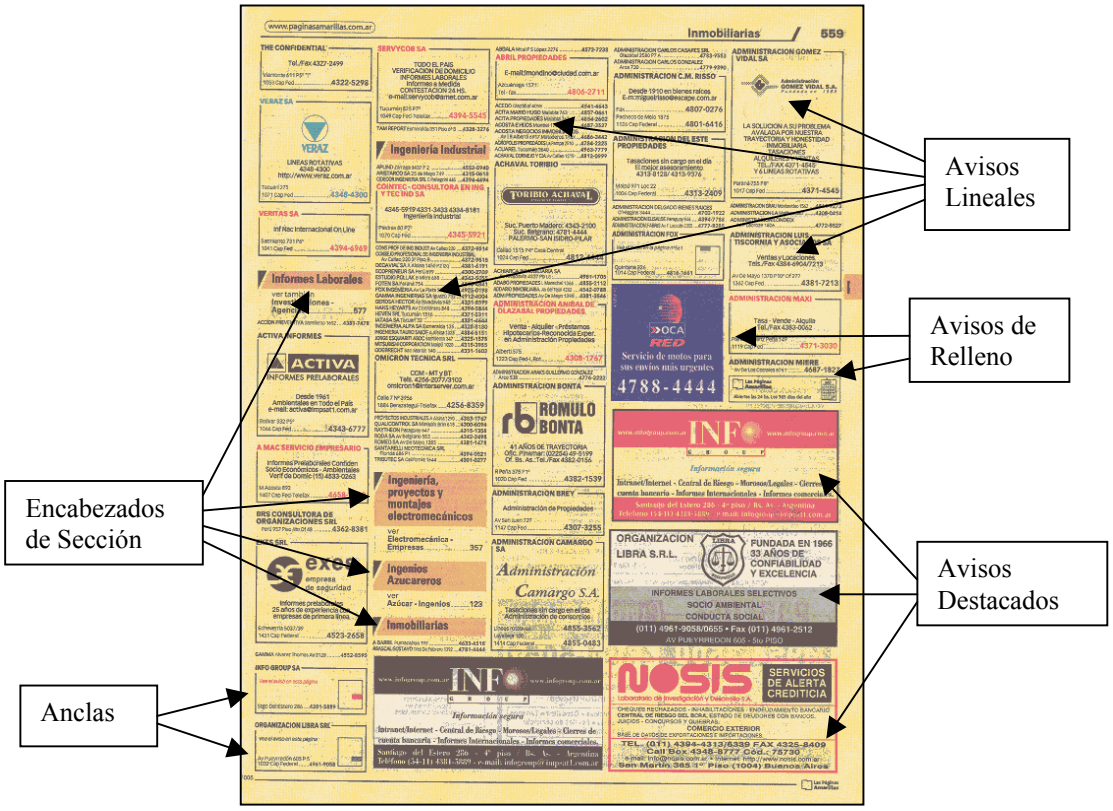

Figura 1: Ejemplo de *Las Páginas Amarillas*

2.2 Criterios generales del Problema de Paginación

Como acabamos de decir, en nuestro trabajo tomaremos como referencia el tipo de layout que se utiliza en los directorios telefónicos de *Las Páginas Amarillas*. En estas publicaciones, los *avisos destacados* y los *lineales* se disponen en páginas de cinco columnas, ubicándose los *lineales* por encima de los *destacados*. Los *avisos destacados* deberían estar ubicados en las páginas de su *rubro* (o *sección*) correspondiente. Al mismo tiempo, para una *sección* dada, los *destacados* más pequeños no pueden aparecer en una página anterior a la de un *destacado* de mayor superficie. Por último, en el caso de los *avisos lineales*, hay que evitar que se produzca un "corte de columna" inmediatamente después de un *encabezado de sección*.

En resumidas cuentas, deben considerarse los siguientes conceptos:

- Cada *página* está subdividida en un número fijo de filas y columnas. La intersección de una fila y una columna determina un *módulo*.
- Cada aviso es de un determinado tipo: *lineal*, *destacado* o *relleno*.
- Cada aviso pertenece a una *sección* en particular (a excepción de los *rellenos*).
- Los *avisos lineales* y los *encabezados de sección* tienen una columna de ancho y un número entero de módulos de alto.
- Los *avisos destacados* pueden tener como mínimo una columna de ancho y como máximo el ancho de la *página*. Las alturas de estos tipos de avisos están definidas como un número entero de módulos.
- Los avisos pertenecientes a un *rubro* dado deben ubicarse "bajo el alcance del *encabezado de sección*" correspondiente. Esta restricción determina que cualquier aviso debería estar en la página de inicio o de fin de sección, o en una página entre ellas.
- Los avisos, preferentemente, no deben mezclarse con los pertenecientes a otras *secciones*.
- Los *avisos lineales* deben, en general, ubicarse en un determinado orden (alfabético).
- Los *avisos destacados* más grandes deben ubicarse lo más cerca posible del borde inferior de la página, y del comienzo de *sección*.
- No se permite ubicar un *encabezado de sección* si no existe al menos un *aviso lineal* debajo de él.
- En cada *página*, los *avisos lineales* deben ubicarse por encima de los *avisos destacados*.

Por otra parte, los datos de entrada del problema estarían dados por:

- *Avisos destacados*: identificador, dimensiones y rubro al que pertenece.
- *Avisos lineales*: identificador, cantidad de módulos (o filas) y rubro.
- *Espacio reservado (o fillers "fijos")*: identificador y dimensiones.

Tanto los *avisos destacados* como los *espacios reservados* pueden, eventualmente, tener una posición predefinida dentro de la publicación (en términos de página y/o posición en la misma).

La salida del mismo, estará constituida por la especificación, en forma precisa, del lugar (número de página, fila y columna) que deberá ocupar cada aviso en el "prearmado" de los clasificados.

2.3 Otras consideraciones

Existen adicionalmente otras cuestiones referidas, principalmente, a los métodos de impresión y a la forma en que se distribuyen los avisos en las páginas de la publicación. Esto complica aún más la problemática planteada.

2.3.1 Estilos de layout

Desde el punto de vista estético, existen ciertos criterios que deben tenerse en cuenta. Uno de ellos establece la forma en que se ubican o agrupan los *avisos destacados* en la página. Dependiendo de la publicación, puede tener preferencia una distribución en forma "piramidal" y con los *destacados* ubicados con mayor densidad hacia el borde exterior de la página. En otros casos, podría priorizarse una distribución en forma de "bloque", manteniendo lo más horizontalmente posible el límite de apilamiento de *avisos destacados*.

Otro criterio que podría considerarse, es el de generar una distribución uniforme de *avisos destacados* y *lineales* a lo largo de la publicación. En este sentido, se introduce el concepto de "umbral", que estaría representando un límite virtual (en términos de superficie) que no debería ser superado al ubicar los *avisos destacados* en la página. El resultado de la fijación de estos *umbrales* tendería a evitar, por ejemplo, que una página dada tenga un gran bloque de *destacados*, mientras que otra posea sólo *avisos lineales*. Sin embargo, siempre estará latente la posibilidad de que una determinada *sección* no tenga *avisos destacados* para ubicar, en cuyo caso sería inevitable una distribución "poco uniforme". Lo mismo podría ocurrir, a nivel global, si se considera que determinadas *secciones* de la publicación tienen una mayor cantidad de *avisos destacados* que otras.

2.3.2 Pliegos

Un *pliego* está formado por una cantidad fija de páginas que se pueden imprimir simultáneamente. Desde el punto de vista del método de impresión, se suele procesar un número exacto de *pliegos*. Este puede variar de imprenta en imprenta, siendo en general, un múltiplo de cuatro (4 páginas por *pliego*, 8 páginas por *pliego*, etc.). La posibilidad de contar de antemano con un "prearmado preciso" de los clasificados, permitiría al departamento de producción determinar la cantidad de páginas extras a agregar (o páginas sobrantes a eliminar) para completar un número exacto de *pliegos*.

En este sentido, la distribución de los avisos (*destacados*, *lineales* y de *relleno*) deberá realizarse de forma tal que hasta el último "rincón" (de la última página del *pliego*) quede cubierto. No se puede entonces hablar necesariamente de la minimización de la cantidad de páginas, ya que por el contrario, podría ser necesario "estirar" la distribución de avisos para alcanzar una cantidad exacta de pliegos.

2.4 Objetivos planteados

Existen diversos objetivos a optimizar que pueden considerarse para el **Problema de Paginación**, en los términos planteados. Cualquiera sea el objetivo definido (podría considerarse sólo uno, o más de uno simultáneamente), siempre deberán tenerse en cuenta los criterios y convenciones generales antes detallados. El estudio preliminar del problema permite identificar, en primera instancia, las siguientes metas:

- Minimización de la superficie cubierta por *avisos de relleno*.
- Minimización de la cantidad de páginas.

La minimización de la superficie cubierta por *avisos de relleno*, permitiría contar con un mayor espacio para la ubicación de avisos pagos en las páginas de la publicación. Sin embargo, y dependiendo del tipo de publicación que se esté analizando, puede haber una restricción en cuanto al número mínimo de *rellenos* a ubicar (por *sección* o en toda la publicación), por ser los mismos una forma de publicidad. Existiría en este caso, una cota inferior para la optimización.

A su vez, la minimización de la cantidad de páginas está estrechamente relacionada con la minimización de la superficie de *relleno*, ya que el hecho de que la superficie de *avisos destacados* y *lineales* sea fija,

determina que la forma de reducir la cantidad de páginas se limite a minimizar la superficie de desperdicio.

Además, como ya dijimos, deberán considerarse ciertos aspectos del layout que hacen a la calidad estética de la publicación.

3 BUSQUEDA TABU

En este capítulo introducimos algunas nociones generales, respecto a la forma de encarar la solución de *Problemas de Optimización Combinatoria* que sean *NP-hard*. En este sentido, y como punto de partida, incluimos una breve descripción de los *Métodos Descendentes* generales. Estos nos servirán de base para la presentación de otra clase de algoritmos más robustos, conocidos como *metaheurísticas*. En particular, centraremos nuestra atención en la *Búsqueda Tabú (BT)*, que fue el elegido para encarar el **Problema de Paginación** en nuestro trabajo.

3.1 Nociones básicas de la Optimización Combinatoria

Muchos problemas de la vida real pueden modelarse como *Problemas de Optimización Combinatoria*. Entre los problemas teóricos clásicos más conocidos de esta clase, podemos mencionar los siguientes: coloreo de grafos, scheduling de tareas, viajante de comercio, bin-packing, entre otros. Como aplicaciones prácticas, encontramos por ejemplo: diseño de redes tolerantes a fallas, problemas de cortes irregulares, diseño de circuitos integrados, construcción de estaciones espaciales, ruteo de vehículos, etc. En particular, como ya dijimos en el Capítulo 1, el problema que nos interesa (**Problema de Paginación**) pertenece a la clase *NP-hard* (ver [JMPS97]), es decir, no se conoce *algoritmo polinomial exacto* para resolverlo.

Algunos de estos problemas pueden ser resueltos simplemente probando todas las combinaciones posibles, pero este enfoque puede ser aplicado solamente a problemas sencillos o con pocos datos de entrada. A medida que la cantidad de datos crece, la cantidad de combinaciones se incrementa vertiginosamente.

Dada la complejidad intrínseca de este tipo de problemas, la búsqueda de "la mejor solución" se torna computacionalmente intratable mediante la utilización de algoritmos exactos o métodos exhaustivos (como aclaramos antes, especialmente cuando los mismos están constituidos por numerosos datos de entrada). Este tipo de *Problemas de Optimización Combinatoria* podríamos abordarlos desde diversas perspectivas. Entre las más comunes, podemos mencionar: el uso de técnicas *Branch & Cut*, y el empleo de *heurísticas de tiempo polinomial determinístico*.

3.2 Métodos de Búsqueda Descendente y Técnicas Heurísticas

A modo de introducción, presentamos aquí las características generales de los denominados *Métodos Descendentes*.

Los *Métodos de Búsqueda Descendente* se basan en que una *solución* dada, puede ser mejorada aplicándole sucesivamente pequeños cambios. El conjunto de soluciones "alcanzables" a partir de una solución, mediante la aplicación de una cambio (o *movimiento*), es denominado *conjunto de vecinos*. Seleccionando sucesivamente la mejor *solución* entre estos *vecinos*, se va mejorando el valor de la *función objetivo*, paso a paso. La *condición de parada* para este tipo de algoritmo suele cumplirse cuando ninguno de los *vecinos* puede mejorar el valor de esa *función objetivo*.

El problema principal de esta estrategia, en su esquema básico, es que resulta "incapaz" de sortear los *óptimos locales* que puedan aparecer. De hecho, su aplicación en la práctica muchas veces no conduce a buenos resultados.

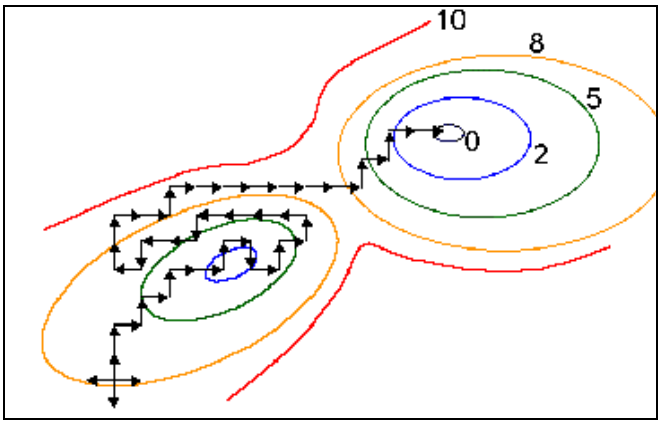

Figura 2: Movimientos de no mejora, para escapar de óptimos locales

Para poder "escapar" de estos *óptimos locales* es necesario permitir, eventualmente, *movimientos de no mejora,* que guiados por alguna *heurística* puedan llevarnos, dentro del *espacio de soluciones,* a regiones más promisorias. A su vez, el permitir *movimientos de no mejora*, podría dar lugar a la generación de *ciclos* en el camino a la *solución*, si no se toman los recaudos necesarios. Mediante la utilización de *técnicas heurísticas* es posible obtener "buenas" soluciones, aunque no necesariamente óptimas, del problema a tratar. En este sentido, recurriendo a estas técnicas es posible acotar el espacio de búsqueda de alguna manera "inteligente".

3.3 Metaheurísticas

Existen ciertas técnicas de mejoramiento (también denominadas *metaheurísticas*) que sirven a su vez, de guía a las *heurísticas locales* tradicionales. La efectividad de estos métodos depende principalmente de su habilidad para:

- escapar de óptimos locales,
- explorar la estructura básica del problema, y
- adaptarse a implementaciones particulares.

Algunas de estas técnicas manejan nociones tales como la *intensificación* y la *diversificación* de la búsqueda. Otras, por su parte, están dotadas de cierta componente *aleatoria*, con el objeto de orientar la búsqueda a otras regiones del *espacio de soluciones*.

Diversas *metaheurísticas*, que sirven de guía a los *Métodos Descendentes* y *heurísticas locales* subyacentes antes caracterizadas, fueron concebidas con el propósito de explotar estos conceptos. De éstas, las más promisorias resultan ser:

- Búsqueda Tabú
- Algoritmos Genéticos
- Simulated Annealing
- Grasp

3.4 Introducción a la Búsqueda Tabú

De entre las *metaheurísticas* recién mencionadas, nos centraremos en la de *Búsqueda Tabú*, que fue la que decidimos utilizar para encarar el **Problema de Paginación**, en nuestro trabajo. Presentamos sus

principales características, términos y posibles extensiones que tornan más robusta su implementación. En este sentido, este punto no intenta ser un tutorial de *Búsqueda Tabú*, sino que tiene por objetivo el situarnos en su marco teórico.

3.4.1 Nociones básicas

Búsqueda Tabú [G95, GL93, L94, L95] es un procedimiento iterativo para encarar *Problemas de Optimización Combinatoria*. Junto a otros métodos de estas características (como *Simulated Annealing*, *Algoritmos Genéticos*, aplicaciones de *Redes Neuronales* y *GRASP*, entre otros), se encuadra dentro de las denominadas *metaheurísticas*. Con antecedentes a comienzos de los años 70, la *Búsqueda Tabú* fue propuesta en su forma actual en 1986 por Fred Glover [G86].

Se ha aplicado con éxito para obtener soluciones óptimas (o aproximadas) a problemas teóricos clásicos como: coloreo de grafos, scheduling de tareas, viajante de comercio, entre muchos otros. Al mismo tiempo, su rápida difusión y crecimiento, en los últimos años, se debe a su exitosa implementación en aplicaciones prácticas: telecomunicaciones, planeamiento de inversiones, asignación de recursos, entre otras.

La filosofía de *Búsqueda Tabú* se basa, principalmente, en los conceptos de *memoria adaptativa* y de *exploración sensible*. Respecto a la idea de poseer "*memoria*", se presenta en oposición a los diseños "sin memoria", basados en ideas tomadas de la física y la biología. Además, el concepto de "*adaptativo*", se contrapone a los métodos catalogados de poseer "memoria rígida", como ser los *Métodos Branch & Bound* y sus variantes relacionadas a la inteligencia artificial. Al mismo tiempo, lo de "*exploración sensible*" hace referencia a la suposición de que "una mala elección estratégica" puede proveer mayor información que "una buena elección aleatoria".

La idea básica del método, descripta por Glover, Taillard y De Werra [GTD93], es explorar el *espacio de las soluciones* mediante una secuencia de *movimientos*. Un *movimiento*, a partir de una *solución* dada, genera otra *solución* que forma parte de su *conjunto de vecinos*. Sin embargo, para escaparse de soluciones óptimas locales, y para evitar el completar un *ciclo* en una iteración particular, se clasifica a ciertos *movimientos* como prohibidos o *tabú*. Estas "prohibiciones", lejos de ser definitivas, se mantienen durante cierto número de iteraciones (*retención* o *tabú tenure*). Al mismo tiempo, *Búsqueda Tabú* contempla la posibilidad de realizar *movimientos de no mejora* como mecanismo para "cruzar" las fronteras de *optimalidad local*, también conocidas como "*barreras*".

Los *movimientos tabú* se basan en la historia de corto y largo plazo de la secuencia de movimientos. Una puesta en práctica simple, por ejemplo, podría clasificar un *movimiento* como *tabú* si el *movimiento* reverso se ha hecho recientemente o con frecuencia. A veces, cuando se juzga "favorable" un *movimiento* clasificado como *tabú*, esta condición puede ser ignorada. Tales excepciones, denominadas *criterios de aspiración,* pueden por ejemplo aplicarse en el caso que un movimiento prohibido conduzca a la mejor solución obtenida hasta el momento.

La *Búsqueda Tabú* se suele aplicar a *Problemas de Optimización Combinatoria*. Supongamos que tenemos una *función objetivo* que devuelve valores numéricos, y se requiere encontrar una *solución* tal que tenga valor mínimo. Como comentamos anteriormente, para los *problemas NP-hard*, este requisito necesita ser relajado a "encontrar un valor cercano al óptimo". Esto es porque cualquier algoritmo exacto conocido requeriría, para determinar la solución óptima, tiempo exponencial respecto del tamaño del problema.

En el caso que podamos relajar la condición a encontrar una "buena" solución, se puede establecer como *criterio de parada* un cierto umbral para la *función objetivo*. O bien, cuando un cierto número de iteraciones se haya cumplido.

Más formalmente,

> Dado S, un conjunto de *soluciones factibles* de una instancia de un *problema de optimización combinatoria*. Se define como *vecindad* de S al conjunto N(S), tal que todas las soluciones de N(S) puedan ser obtenidas a partir de aplicar una *función de movimiento* válida a S, según la *heurística* subyacente.

El método comienza con una *solución inicial* i, posiblemente generada al azar y no siempre factible, a partir de la cual se genera una secuencia de *soluciones*. En cada iteración se selecciona la *vecindad*. El proceso de *selección* es el primer paso para determinar los *vecinos* de i, y el conjunto *aspirante de vecinos*. A partir del primer *movimiento* realizado, se comienza a "poblar" la *lista tabú*. Evitar ciclos es uno de los principales objetivos al restringir ciertos *movimientos*. En consecuencia, el criterio más utilizado es el de caratular los *movimientos reversos* como *tabú*.

En este punto, la *vecindad* original se modifica, descartando aquellas *soluciones* que hayan sido catalogadas como *tabú*.

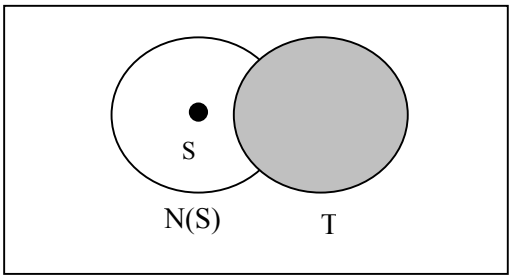

Figura 3: Vecindad de una Solución y conjunto de Movimientos Tabú

En algunos casos en que la cantidad de *vecinos* sea muy grande, se puede utilizar una *función aleatoria* (en general, con distribución uniforme), para generar un subconjunto de N(S). De esta forma, se reduce la cantidad de *vecinos* a analizar.

Algoritmo básico de Búsqueda Tabú:

1) Elegir una solución inicial i; i*=i; k=0.
2) k=k+1; Generar un subconjunto N* de N(i,k), tal que no cumpla las condiciones tabú, o cumpla el criterio de aspiración.
3) Elegir el mejor j ϵ N* con respecto a f(j), y hacer i=j.
4) Si f(j) < f(i*), hacer i*=j.
5) Actualizar la lista tabú y condiciones de aspiración.
6) Si no se cumplen las condiciones de parada, volver a (2).

En el pseudocódigo observamos cómo se trata de evitar "estancarse" en los *óptimos locales*, mediante la selección de los *vecinos* en N*, dejando fuera del análisis aquellos *movimientos* que están marcados como *tabú* (a excepción de los que cumplen el *criterio de aspiración*). Las condiciones para que un *vecino* sea *tabú* o *aspirante* serán específicas según el dominio del problema. Por ejemplo, un *movimiento* puede ser clasificado como *tabú* si pudiera conducir a una *solución* que se ha considerado ya en las iteraciones

pasadas (*recency* o memoria de corto plazo). Otra alternativa podría ser la de prohibir un *movimiento*, si el mismo conduce a una *solución* visitada muchas veces (*frecuency* o memoria de largo plazo). Por otro lado, un *movimiento tabú* satisface los *criterios de aspiración* si, por ejemplo, el valor de la *solución* generada es mejor que el mantenido hasta el momento.

3.4.2 Posibles extenciones

La *Búsqueda Tabú* tal como acabamos de presentarla, no siempre produce los mejores resultados. Dependiendo del problema, a veces es necesario aplicar otras técnicas complementarias con el objeto de profundizar (*intensificar*) y/o extender (*diversificar*) la búsqueda, respecto del universo de soluciones.

Diversificación

Es importante *diversificar* la búsqueda respecto del *espacio de soluciones*, ya que de no hacerlo, podrían quedar grandes regiones del mismo sin ser exploradas. Una forma general de ampliar o extender la búsqueda podría ser la de, cada cierta cantidad de iteraciones, forzar un reinicio del algoritmo, y partir de una nueva *solución inicial* generada al azar. La componente aleatoria no nos garantiza que todas las regiones se exploren, pero existe una gran posibilidad de pasar de una región a otra.

Otra manera de *diversificar* la exploración, pero más relacionada con la metaheurística de la *Búsqueda Tabú*, es la de incrementar la *retención* de un *movimiento* en la *lista tabú*. Los movimientos que se marcan como prohibidos, serán mantenidos en esa condición por un cierto período de tiempo. Cuanto más alta sea esa retención impuesta, hay más probabilidades de que las soluciones generadas pertenezcan a otras regiones, ya que todas las posiciones cercanas pueden estar vedadas.

Por otro lado, además es posible armar la *lista tabú* en base al concepto de *frecuencia*. Si una solución es visitada muy frecuentemente, el prohibirla es otra buena táctica para diversificar la búsqueda.

Intensificación

En contrapartida a la *diversificación*, la *intensificación* intenta obtener la mejor solución, analizando en profundidad una región determinada. El modificar las reglas de *selección* de elementos, alentando la combinación de atributos de buenas soluciones históricamente encontradas, puede resultar una buena estrategia para intensificar una búsqueda.

Una manera sencilla de hacerlo, es reducir el tamaño de la *lista tabú*. De esta manera, si bien se corre el riesgo de producir ciclos, se asegura que una región sea explorada intensivamente.

Normalmente, *intensificación* y *diversificación* se combinan entre sí, con algunas técnicas auxiliares, para producir mejores resultados.

En este sentido, una forma sencilla de combinar ambas técnicas consiste en variar dinámicamente la *retención* de la *lista tabú*. Se puede hacer que el algoritmo vaya alternando períodos de *diversificación* e *intensificación*, simplemente incrementando o decrementando dicho valor cada cierto período de tiempo. De esta manera, se puede decir que el *espacio de soluciones* será recorrido extensivamente, y con ciertos períodos intermedios en los que se profundizará la búsqueda. Otra alternativa para variar la *retención*, es seleccionar al azar este valor cada vez que se incluye un nuevo elemento en la *lista tabú*. De esta manera, introduciendo cierta componente aleatoria, también es posible combinar la intensificación y la diversificación.

Lista de Soluciones Elite

Las *soluciones elite* son aquellas "buenas" *soluciones* que han sido generadas en algún momento durante la corrida del algoritmo. Normalmente estas soluciones se detectan en las etapas de *diversificación* de búsqueda, y se las almacena en una lista, para su posterior análisis en profundidad (*intensificación*). Así, combinando también *intensificación* y *diversificación*, tenemos una alta probabilidad de encontrar la *solución* óptima (o eventualmente, una suficientemente buena).

4 ALGORITMO PROPUESTO

En este capítulo explicamos las distintas heurísticas propuestas. Las mismas abarcan la construcción de la *solución inicial*, la ubicación de los avisos en la página, el particionamiento en ventanas de optimización, y los movimientos factibles de realizar en la *Búsqueda Tabú*. Posteriormente, presentamos cómo se podría aplicar la técnica de *Búsqueda Tabú* al problema que tratamos.

4.1 Heurísticas propuestas

Tal como hemos dicho, nosotros encaramos el **Problema de Paginación** desde el *enfoque heurístico*. En algunos casos, tomamos ideas generales recolectadas en el relevamiento bibliográfico realizado, y las adaptamos a nuestro problema particular. Inclusive, en algunos de ellos, introdujimos mejoras a las heurísticas encontradas. En otros casos, las heurísticas desarrolladas (y posteriormente implementadas) provienen de ideas propias.

Como describimos en detalle a continuación, desarrollamos y analizamos diversas heurísticas, focalizadas en diferentes componentes de la *Búsqueda Tabú* y en problemáticas específicas de la Paginación: la *solución inicial*, los *movimientos de mejora*, la *ubicación de avisos en las páginas*, y el tratamiento de la *publicación por ventanas*.

4.1.1 Solución Inicial

La *solución inicial* es la solución a ser mejorada por el proceso de *Búsqueda Tabú*. Dependiendo de la implementación del algoritmo, la calidad de la *solución inicial* puede influir o no en la calidad de la solución final obtenida.

En nuestro trabajo, para generar la *solución inicial*, desarrollamos una *heurística constructiva golosa*. Tomando como punto de partida la idea propuesta por Johari, Marks, Partovi y Shieber (en [JMPS97]), desarrollamos una heurística mejorada para generar una *solución inicial* no necesariamente factible.

En este sentido, y como paso previo, ordenamos los *avisos destacados* por *sección* y, dentro de cada *sección*, por superficie. De esta forma, "forzamos" uno de los requerimientos del problema: que, para una *sección* dada, los avisos más grandes se ubiquen "más cerca" del inicio de *sección* que los avisos más pequeños. Al mismo tiempo, como veremos más adelante, evitamos tener que considerar este criterio en el cálculo de la función evaluación.

En el trabajo mencionado, los autores proponen estimar la cantidad de páginas necesarias para alojar todos los *avisos lineales* y *destacados*, y posteriormente le suman un adicional (entre un 2% y un 6%). A continuación, distribuyen en forma pareja los *avisos destacados* en las páginas, teniendo en cuenta su superficie, y manteniendo el ordenamiento (sección/superficie) de los avisos.

A diferencia de eso, nosotros proponemos considerar además la proporción de superficies de *avisos lineales* y *destacados* para cada *sección*. Con esto buscamos no recargar las páginas de *avisos destacados*, bajando la probabilidad de que se produzca *overflow* en la *solución inicial*.

4.1.2 Heurísticas de Mejora

Las heurísticas de mejora que analizamos para implementar el *movimiento* de una *solución* dada a una *solución* vecina, se focalizaron en producir cambios respecto de los *cortes de página*. En este sentido, consideramos diferentes alternativas:

- Agregar un corte de página (split)
- Eliminar un corte de página (join)
- Cambiar de lugar un corte de página (shift)

Entre estas opciones, nos inclinamos por la de *cambiar un corte de página de lugar*, ya que estimamos que agregar o eliminar cortes de página puede generar soluciones bastante distantes unas de otras.

A su vez, respecto de la alternativa elegida, analizamos dos variantes de *shift*: uno *acotado* y otro *no acotado*.

El *shift no acotado* se basa en eliminar un corte de página de un lugar dado, y en insertarlo en otro cualquiera. A primera vista, se está "saltando" así a soluciones muy distantes.

El *shift acotado*, que consiste en desplazar un corte de página entre los cortes de página adyacentes, resulta en primera instancia la clase de movimiento "más natural". Tendría el efecto de "pasar" *avisos destacados* de una página a otra (pero entre páginas adyacentes).

4.1.3 Ubicación de avisos en la página (Layout)

Respecto de la disposición los *avisos destacados* en una página dada, desarrollamos diferentes *heurísticas constructivas*, de acuerdo a los distintos "criterios" gráficos utilizados con mayor frecuencia en las publicaciones comerciales. Estas heurísticas tienden a distribuir los *avisos destacados* de forma tal que los mismos formen bloques "uniformes", desde el punto de vista gráfico. Las mismas pueden ser vistas como *heurísticas golosas* (sumamente "optimistas"), si se considera que buscan en cada momento la mejor posición espacial de cada aviso tratado.

En este sentido, las diferencias entre los distintos estilos considerados por las heurísticas desarrolladas estarían dados, exclusivamente, por la forma en que se ubican los *avisos destacados* en la página:

- hacia la parte inferior de la página
- hacia la parte superior de la página
- hacia el borde externo de la página
- hacia el borde interno de la página

A su vez, estas heurísticas consideran la posibilidad de agrupar los avisos, generando bloques "apilados" o "acostados" (es decir, agrupándolos vertical u horizontalmente).

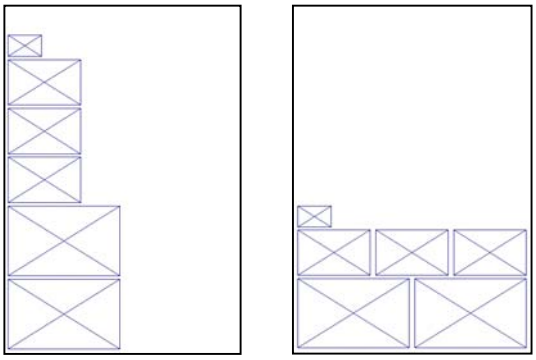

Figura 4: Layout "apilar" y "acostar"

A modo de ejemplo, supongamos que queremos distribuir los *destacados* "al estilo de **Las Páginas Amarillas**". Combinando las opciones de "inferior" y "externa" es posible generar distribuciones de los *avisos destacados* en forma de "U", al mirar de a dos páginas (par/impar) a la vez.

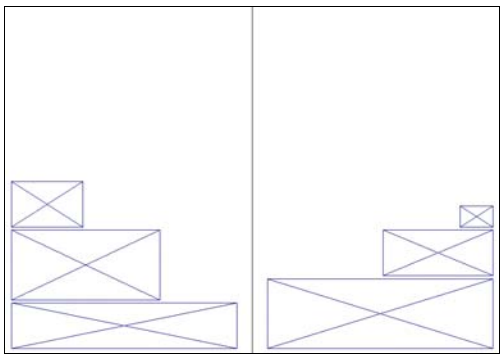

Figura 5: Layout en forma de "U"

Como un paso previo a la ubicación de los avisos en la página, los ordenamos por ancho (y ante igual ancho, por alto), en forma decreciente.

El algoritmo inicia la búsqueda tomando como punto de partida la esquina inferior-externa de la página, e intenta sucesivamente "mejorar" dicha posición, a medida que se desplaza (columna a columna) hacia el borde interno de la página. La idea es obtener de esta forma, una ubicación para el aviso en cuestión lo más cercana posible al borde inferior-externo de la página actual, verificando en cada momento que el aviso a ubicar:

- quepa en la página actual, y
- no "pise" otro aviso previamente ubicado

En el caso de estar agrupando *verticalmente*, el algoritmo conservará la posición más elevada para cada columna analizada. Por el contrario, si está agrupando *horizontalmente*, conservará la de menor profundidad.

A las heurísticas propuestas podríamos incorporarles ciertas mejoras con el "*reconocimiento* de bloques de avisos". Es decir, un conjunto de *avisos destacados* podrían tratarse con un sólo aviso (si, por ejemplo, tienen la misma altura). De esta forma, en determinadas circunstancias, podríamos generar mejores formaciones (eventualmente, con menor cantidad de desperdicio).

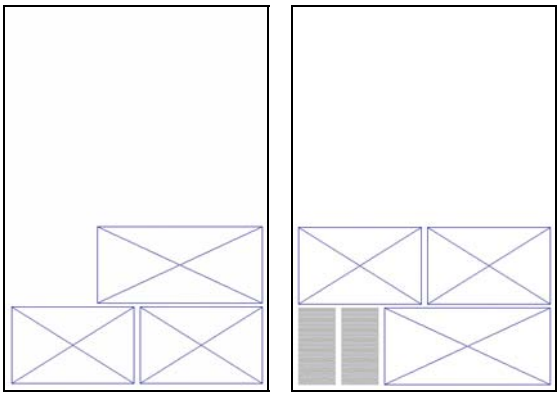

Figura 6: Composición de bloques

Otra mejora factible de realizar, consiste en alterar el orden de entrada de los avisos a ubicar en la página. En la heurística original, los avisos hacían su ingreso en la página, uno a uno, ordenados por ancho (y alto). La mejora consistiría en realizar permutaciones en el orden de entrada, haciendo varios intentos de ubicación de los avisos, y eligiendo la "mejor" combinación generada. De esta forma, estaríamos analizando mayor cantidad de variantes de formaciones.

4.1.4 Manejo de "Ventanas de Optimización"

Una decisión, también "heurística", resulta de partir la publicación a paginar en "Ventanas de Optimización". Esto permite manejar publicaciones de cualquier tamaño, acotando significativamente el espacio de búsqueda.

En este sentido, el algoritmo principal se implementa de manera tal de "solapar" las sucesivas ventanas. Así, por ejemplo, el material (*avisos lineales*, *destacados* y de relleno) de la última página de una ventana sería reprocesado con la siguiente ventana de optimización.

Si bien la partición del problema en ventanas permitiría el procesamiento en paralelo, el solapamiento de las mismas genera cierta dependencia ente ellas, forzando en primera instancia, su procesamiento en forma secuencial.

4.2 Aplicación de "Búsqueda Tabú" al problema de paginación

La implementación que hemos hecho, está focalizada en la aplicación de la *Búsqueda Tabú* al Problema de la Paginación de Avisos Clasificados. Esta implementación nos ha requerido hacer un diseño complejo que será explicado en los siguientes ítems.

4.2.1 Representación de la información

Para los propósitos de la *Búsqueda Tabú*, fue necesario encontrar una representación de los datos sobre la cual fuera posible aplicar esta técnica.

La información a representar es la siguiente:

> *Sección*: Es la entidad que agrupa *avisos lineales* y *destacados*. Como información propia, contiene detalles del encabezado, referencias a los datos, y un resumen acerca de los avisos contenidos.

Destacados: Esta entidad contiene los detalles propios del aviso: ancho, alto y a que sección pertenece. Sobre esta estructura, como se verá mas adelante, actúa la *Búsqueda Tabú*.

Lineales: Estos avisos están referenciados desde las secciones, y se utilizan al momento de ubicarlos en las páginas.

Solución: La solución es la estructura que debe ir mutando con el correr de las iteraciones de la *Búsqueda Tabú*. Como principales características, contiene información relativa a la ubicación de los avisos en las distintas páginas y el desperdicio generado, así como también el valor obtenido en su evaluación.

Lista tabú: En esta estructura, mantenemos la historia de los últimos movimientos realizados. Se utiliza para saber si un posible movimiento es tabú o no.

Movimientos: Es la estructura que representa el movimiento de un corte de página, cuando pasa de un lugar a otro.

Como dijimos anteriormente, los cortes de página ubicados sobre la lista de *destacados*, es lo que determina qué avisos pertenecen a cada página. De esta manera, se simplifican varias de las estructuras de datos: la lista tabú, la solución y los movimientos.

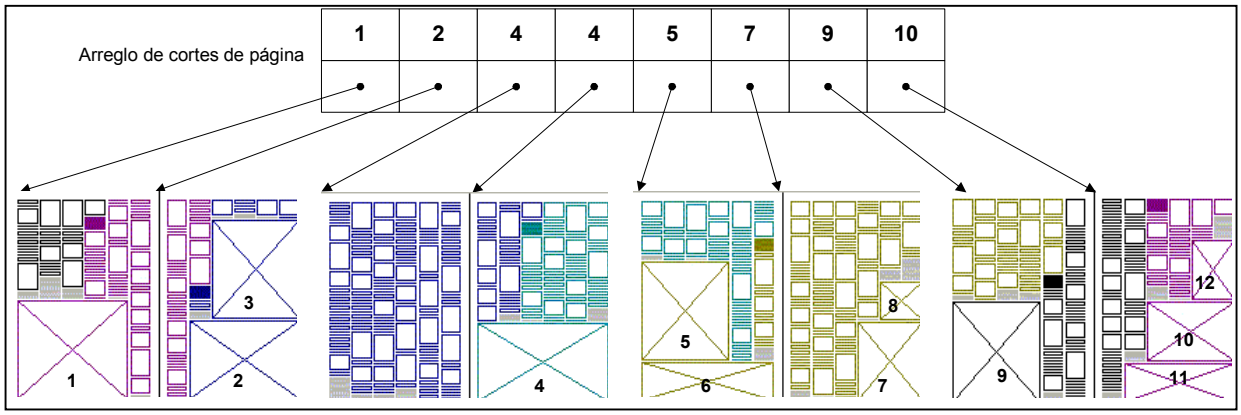

Figura 7: Representación de los "Cortes de Página"

Por otro lado, la premisa utilizada de no alterar el orden de los avisos (vienen ordenados por sección y dentro de ésta por tamaño) es totalmente acorde a esta representación. Esto hace que la lista de avisos se mantenga inalterable y que lo único que deba cambiar es la posición de los cortes de página.

4.2.1.1 Representación de la solución candidata

Como explicamos anteriormente, una solución candidata debe tener información de la ubicación de avisos en las páginas, el desperdicio, y el valor de su evaluación.

Con respecto a la ubicación de *avisos destacados* en las distintas páginas, la estructura solución almacena valores que representan los cortes de página. Estos valores referencian al primer *destacado* de la página siguiente. Teniendo en cuenta que los *destacados* están ordenados en una lista y que este orden no se puede alterar, con estas referencias podemos saber exactamente qué *destacados* corresponden a cada página. También guardamos en la estructura *solución*, una referencia al primer *aviso lineal* que corresponde a cada página, a los fines de tener presente en todo momento qué *lineal* corresponde pegar. Los *avisos de relleno*, se almacenan en la solución, ya que estos objetos no existen como elementos de

entrada, sino que se producen inevitablemente en la distribución de los avisos en las páginas. Por último, el valor de la solución es un atributo propio de la misma, y se calcula luego de haber ubicado todos los avisos correspondientes.

Con respecto a la ubicación de los avisos en las páginas, debido a que contamos con un proceso que trabaja de forma determinística, no es necesario guardar información de posiciones a nivel de la solución. Como se verá más adelante, este procedimiento es capaz de generar la misma disposición simplemente sabiendo cuáles son los *avisos destacados* a ubicar y cuál es el primer *aviso lineal* de la página. Así, con los datos almacenados podemos reproducir la disposición de los avisos sin inconvenientes y las veces que sea necesario.

4.2.2 Generalidades del diseño

Hemos decidido acotar la *Búsqueda Tabú* a determinar qué *avisos destacados* se ubican en cada página. Es decir, para los fines propios del algoritmo, los únicos elementos existentes son este tipo de elementos. De esta manera, simplificamos en parte lo que a simple vista parece un problema complejo. De todas formas, existe una interacción entre los *avisos destacados* y los comienzos y fines de sección, y en consecuencia, también con los *avisos lineales*. Por este motivo, en la calidad de la solución se tienen en cuenta todos los objetos, los cuales necesariamente deben ser tratados por procesos complementarios, invocados desde el cuerpo principal del algoritmo.

4.2.3 Ubicación de avisos en la página

Como hemos anticipado, este algoritmo es un proceso auxiliar al de *Búsqueda Tabú*, y hemos definido su comportamiento de forma determinística.

El proceso actúa página por página, en forma independiente. Lo primero que hace, es ubicar los *avisos destacados*. Luego de esto, fluye los *avisos lineales* correspondientes y calcula los espacios de relleno generados en la tarea.

Este procedimiento es invocado por el algoritmo principal, luego de que *Búsqueda Tabú* define qué *avisos destacados* se ubicarán en cada página. Los datos generados se usan para el cálculo del valor de la solución obtenida.

4.2.4 Componentes de la Búsqueda Tabú

4.2.4.1 Lista Tabú

Existen varias maneras de implementar la lista tabú. Una de las posibilidades analizadas fue la de restringir movimientos teniendo en cuenta el origen y el destino del mismo, es decir, el movimiento opuesto a uno realizado. De esta manera, evitamos que si se realiza un movimiento de un corte de página de la posición 5 a la 7, haya un movimiento reverso de la posición 7 a la 5, evitando de esta manera ciclos. Lo que esta restricción no evitaría, es un ciclo en un potencial tercer paso, es decir, si el corte de página pasa de la posición 5 a la 7, y luego de la posición 7 a la 8, nada le impide moverse de la posición 8 a la 5.

Otra posible implementación, es la de marcar como tabú la ubicación de un corte de página determinado en una posición anteriormente ocupada por él. En este caso, logramos una buena política de restricción de movimientos. La implementación de este esquema requiere de una matriz de D x P, siendo D la cantidad de *avisos destacados*, y P la cantidad de cortes de página del problema. En cada posición guardamos un valor que nos indicará la vigencia de la restricción.

Una tercera opción, sería marcar como tabú una posición determinada. De esta manera, ninguno de los cortes de página podrá ocupar ese lugar. Este esquema es más restrictivo que el anterior, y en la práctica fue el que mejores resultados obtuvo. La implementación del mismo consta de un arreglo de largo D, en el cual, al igual que en el caso anterior, se almacenan datos relativos a la duración de la restricción.

Con respecto a la retención de un movimiento tabú, hemos probado y analizado dos alternativas: una de ellas es la de tener una duración fija, y la otra, de variarla dinámicamente dentro de un determinado rango. La experiencia muestra que el valor óptimo para la retención de un elemento como tabú, depende del tamaño del problema tratado ([L95]). Por esta razón hemos implementado ambas alternativas, corroborando finalmente, los valores teóricos. Para la implementación de la retención variable, se utilizó el método sugerido por F.Glover y M.Laguna en [GL93]. De esta manera, definimos el intervalo [a $\sqrt{d}$, b $\sqrt{d}$], para los valores de retención, siendo "d" la cantidad de *avisos destacados*. Los valores de "a" y "b" han sido ajustados en base a nuestras pruebas.

Otra posible implementación de una retención variable, es hacerla variar cada cierto período de tiempo. De esta manera, el algoritmo tendrá distintas etapas, en donde para valores de retención altos se dedicará a hacer una búsqueda más diversificada por el espacio de soluciones, mientras que para valores de retención bajos, tratará de obtener el valor óptimo de cada valle encontrado. Este tipo de variación de la retención podría ser conveniente usarla en combinación con soluciones elite.

Para simplificar el procedimiento de verificación de si un movimiento es tabú o no, y para permitir manejar retenciones variables, hemos decidido guardar el número de iteración en el cual un elemento deja de ser tabú. En este caso, basta comparar este valor con el del ciclo actual para saber si aún está en vigencia. La opción más natural, la de guardar el ciclo que se marcó como tabú, nos obliga a calcular la diferencia entre este valor y el del ciclo actual, y sólo es aplicable en casos de retención fija.

4.2.4.2 Criterio de Aspiración

En este caso, el más natural, incluso el más referenciado en la bibliografía, es el de tener un criterio de aspiración aplicable sólo en casos en que la solución tabú tenga el mejor valor de evaluación que haya sido obtenido hasta el momento.

Otro posible criterio de aspiración podría haber sido el de comparar el valor de la solución tabú con el de las últimas n iteraciones. En este caso, podríamos cambiar de valle sin obtener una solución mejor que la mínima encontrada hasta el momento, aunque de esta manera se corren riesgos de producir los indeseados ciclos.

Hemos elegido el primer caso porque es el más popular.

4.2.4.3 Criterio de Parada

Los criterios de parada evaluados para la detención del algoritmo han sido: el número máximo de ciclos, el número máximo de ciclos sin obtener mejora, y el alcanzar un valor de solución razonablemente bueno. Consideramos suficiente manejar las paradas por cantidad total de ciclos y por un valor mínimo de evaluación.

4.2.4.4 Función "Movimiento"

La función movimiento es la que permite pasar de una solución dada a una solución vecina.

Elegimos definir el movimiento en términos de "cambiar un corte de página de lugar" (shift). En este sentido, como ya mencionamos, tenemos a su vez opciones:

- Movimiento acotado
- Movimiento no acotado

El movimiento acotado, permitiría mover un corte de página no más allá del corte de página siguiente o anterior, afectando de esta manera, sólo a dos páginas por movimiento. Este movimiento es el más natural, ya que es el equivalente de traspasar avisos de una página a la anterior o a la siguiente.

El movimiento no acotado, causaría un efecto equivalente al de eliminar un corte de página (join) en un lugar, y al de insertar un corte de página (split) en otra parte de la publicación, generando así soluciones muy distantes.

En el caso de que no haya suficiente cantidad de páginas para los avisos procesados, lo que hacemos es agregar páginas al final de la publicación. El algoritmo irá incrementando la cantidad en base a lo que necesite.

4.2.4.5 Lista de Movimientos Candidatos

El conjunto de movimientos candidatos definen, a partir de una solución dada, el vecindario de la misma. Como acabamos de explicar, consideramos los movimientos de cortes de página de un lugar a otro (shift). En este sentido, a continuación analizamos cotas superiores de las dos variantes de movimiento propuestas: del shift acotado y del shift no acotado.

Para el caso del "no acotado", la cantidad de vecinos máxima que podemos encontrar, es P x (D-1), siendo P la cantidad de cortes de páginas, y D la cantidad de *avisos destacados*. Esto es debido a que cualquier corte de página puede moverse a cualquier posición en la lista de *avisos destacados*, excepto a la posición que ocupa en este momento (de ahí el D-1). Si quisiéramos hacer un ajuste más fino, en esta multiplicación habría que restarle a P la cantidad de cortes de página ubicados en la misma posición, ya que por ejemplo, si P_3 y P_4 ocupan la posición 6, es indiferente que se moviera P_3 o P_4 a la posición 9. Si llamamos P' a la cantidad de cortes de página que ocupan distintas posiciones (o la cantidad de posiciones distintas ocupadas por cortes de página), la fórmula final para calcular la cardinalidad del vecindario sería P'x (D-1)

Para el caso de los movimientos "acotados", el número de vecinos que se pueden generar es sustancialmente menor. La cota máxima de movimientos es 2 x D (siendo D la cantidad de *destacados*). Para comprender el por qué, analicemos el problema dividido en dos partes. Contemos primero los cortes de página que pasan de D_i a D_j siendo $i<j$ (cortes de página que se mueven hacia la derecha). Supongamos que P_0 ocupa la posición 0. P_0 puede moverse hasta la posición que ocupa P_1 inclusive. P_{i-1} puede moverse n lugares hasta la posición de P_i. En definitiva, cada posición destino solamente puede ser alcanzada por el corte de página inmediatamente anterior. De esta forma, como los lugares para ubicar cortes de página son D, y como cada posición puede ser alcanzada por un solo corte de página, la cantidad de movimientos disponibles es D.

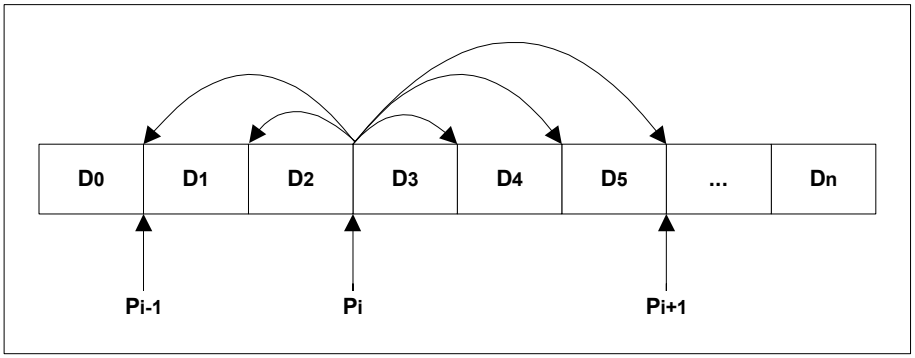

Figura 8: Movimiento "acotado"

Para complementar el 100% de los movimientos, debemos considerar además los que van hacia la izquierda, y aplicando el mismo razonamiento, obtenemos D como movimientos disponibles. En consecuencia, el máximo número de vecinos del movimiento "acotado" se reduce a 2 x D. El número exacto de vecinos, puede diferir de este valor en el caso de que P_0 no esté ubicado en la posición 0, o que el último corte de página no esté en la posición D. En estos casos, a 2 x D hay que restarle la distancia entre 0 y la posición de P_0 y la distancia entre D y la posición del último corte de página.

Con respecto a la cantidad de vecinos, podemos ver que es una cantidad considerable, especialmente en la primera opción. Para limitar el espacio de búsqueda del algoritmo, se pensó en elegir una cantidad determinada del total, configurada a través de parámetros, que podría ser una cantidad fija o un porcentaje del total de movimientos. Se eligió esta última opción porque es más razonable pensar en una cantidad que esté relacionada con el tamaño del problema, y no en un número que puede resultar absolutamente arbitrario. Estos vecinos son seleccionados del conjunto total, en forma aleatoria, utilizando una distribución uniforme.

4.2.4.6 Función Evaluación

La función evaluación tiene en consideración los siguientes criterios:

- Overflow: Cuando dos o más avisos que pertenecen a una misma página, no pueden ser ubicados de manera que no se superpongan. Este defecto se penaliza duramente, ya que no es admisible que exista en una solución final.
- Aviso fuera de orden: Para una sección determinada, el hecho que un *aviso destacado* de menor tamaño ocupe una página anterior a un aviso mayor, debe ser penalizado. El criterio de penalización toma en cuenta la cantidad de avisos en estas condiciones.
- Distancia al encabezado de sección: Se suma al valor de la solución un valor que depende de la distancia que hay entre los *avisos destacados* y el *encabezado de sección*. A mayor distancia, mayor penalización.
- Aviso previo al encabezado de sección: Se penaliza de manera especial si un *aviso destacado* se ubica en páginas anteriores a su correspondiente *encabezado de sección*.
- Superficie de relleno: Se penaliza la generación de rellenos y *burbujas*. De esta manera, se busca minimizar la cantidad de páginas utilizadas.
- Preferencia par o impar: Si existen avisos con preferencia determinada, se penaliza el caso de que esta preferencia sea violada.

La última página de la publicación necesita un tratamiento especial, ya que por ejemplo, no se debe penalizar el área de desperdicio debido a que ésta se genera por la finalización de los datos de entrada.

4.2.5 Parametrización

De acuerdo al análisis detallado que vinimos haciendo, los parámetros a considerar en la implementación podrían ser divididos en tres grandes categorías: parámetros propios de la *Búsqueda Tabú*, parámetros de la Función Evaluación, y parámetros referidos al Layout de páginas.

Dentro de la primera categoría, los parámetros se refieren a la *cantidad máxima de ciclos* y a un posible *valor mínimo de la solución*, para ser utilizados como criterio de parada. Otra variable a ajustar, es la *cantidad de vecinos* a procesar del total de la lista de candidatos. Además, podemos ajustar los valores referentes a la *retención* de los elementos en la lista Tabú.

Dentro de los parámetros de la Función Evaluación, tenemos todos aquellos valores utilizados para penalizar los distintos criterios considerados. Cada criterio tiene su valor de evaluación asociado separado de los demás, de forma tal que se puedan manejar todos ellos de manera independiente.

En cuanto al tercer grupo, los de Layout, permiten definir el tamaño de las páginas (en términos de filas y columnas), así como también algunas cuestiones de estilo de distribución de los *avisos destacados*.

5 IMPLEMENTACION

La implementación del prototipo fue dividida en dos grandes módulos: *Motor* e *Interfase Gráfica*. A grandes rasgos, el *Motor* abarca todo lo referente a procesamiento de *Búsqueda Tabú*, mientras que la *Interfase Gráfica* (también denominada *Interfase de Usuario*) fue pensada para simplificar la tarea de compaginación y visualización de la publicación. Desde la *Interfase Gráfica* se puede abrir una publicación, configurar parámetros, personalizar algunos criterios en cuanto a la ubicación de los avisos, invocar al *Motor*, y visualizar los resultados. Mediante este esquema, podemos decir que tenemos una aplicación funcional y sencilla de utilizar, que cubre en gran parte de los requerimientos planteados.

En esta etapa se introduce el concepto de *ventanas de optimización*, mediante el cual los datos de entrada son particionados para facilitar su procesamiento.

5.1 Motor

Esta componente fue desarrollada en C++, y se encarga de ubicar los avisos en las distintas páginas. En este módulo, implementamos la partición en *ventanas* y la *Búsqueda Tabú*, lo que hace de él el punto central de nuestro trabajo.

5.1.1 Ventanas de Optimización

Los datos de entrada no se procesan todos juntos, sino que se particionan en *ventanas de optimización* para ser procesados de forma más efectiva. La *ventana* consiste de un número configurable de páginas, lo que permite que la información a procesar se mantenga en un volumen manejable para la *Búsqueda Tabú*.

Cada corrida del algoritmo consta de N *ventanas*, que se procesan secuencialmente. Las *ventanas* tienen cierta dependencia entre sí, que consiste en que la cola de una de ellas se reprocesa junto con la siguiente. Por esta razón, se dificulta implementar un procesamiento en paralelo. Esta superposición de las *ventanas* es llamada "*solapamiento*", de aquí en adelante. En este sentido, tanto la cantidad de páginas del *solapamiento*, como así también el tamaño de la *ventana* son parametrizables.

El manejo de *ventanas* comienza en el momento en que se cargan los datos a memoria, levantando sólo los avisos necesarios para cubrir la superficie de una *ventana*. Los avisos restantes son en principio ignorados, y serán procesados en las *ventanas* subsiguientes. Con estos datos, se invoca al procedimiento de *Búsqueda Tabú*, el cual arrojará resultados para la cantidad de páginas definida en la *ventana*. Los datos que no pudieron ser ubicados dentro de estas páginas (hayan sido levantados a memoria o no) serán procesados con la siguiente *ventana*, junto con los avisos ubicados en las páginas del *solapamiento*.

5.1.2 Estructuras de datos

En este punto, implementamos las estructuras definidas en el capítulo anterior, conjuntamente con otras estructuras auxiliares utilizadas para el manejo de *ventanas de optimización*. También implementamos otras estructuras auxiliares para cubrir algunos detalles no contemplados hasta este momento.

> ***Secciones*:** Fueron implementadas como una lista global en memoria. Cada elemento de la lista contiene: información del *encabezado de sección*, referencias a los *avisos lineales* del rubro, y valores sumarizados de cantidad y superficie de avisos (diferenciando por tipo de aviso). Los datos aquí contenidos son levantados a memoria al comienzo del algoritmo principal, y no son alcanzados por la partición en *ventanas*. Además, en esta estructura almacenamos los valores de página, fila y columna asignados al *encabezado de sección*, en la solución final.

***Avisos destacados*:** Los *avisos destacados* también fueron implementados como una lista global. A diferencia de la lista de *secciones*, esta sí es dependiente de la *ventana* en proceso. Con cada comienzo de *ventana* se leen los archivos de entrada, levantando a memoria aquellos datos que estarán involucrados en la *ventana* actual. Existen dos listas de *avisos destacados*: una para los avisos con página prefijada por el usuario (*avisos destacados fijos*), y otra, para aquellos avisos sin este tipo de predeterminación (*avisos destacados flotantes*). Como se verá más adelante, ambos tipos de objetos tienen un tratamiento distinto, y es por eso que han sido separados en dos listas independientes. Para cada elemento de estas listas, almacenamos información propia del *aviso destacado*, como ser: *rubro*, dimensiones (en filas y columnas), si tiene o no una posición predeterminada (dentro de la página), y también si tiene preferencia de página par o impar. En el caso de los *avisos destacados fijos*, también figura el número de página preasignado.

***Avisos lineales*:** Los *avisos lineales* de cada *rubro* están implementados como arreglos dinámicos. Como antes mencionamos, estos avisos están referenciados en la lista de *secciones*. Los datos que contiene cada elemento se refieren a: tamaño del aviso (en cantidad de filas), y la posición (página, fila y columna), que es asignada en el momento de generar la solución final.

***Solución*:** Esta estructura tiene como principal componente, un arreglo de *cortes de página*, los cuales hacen referencia a la lista de *destacados flotantes* para indicar a qué página corresponde cada aviso.

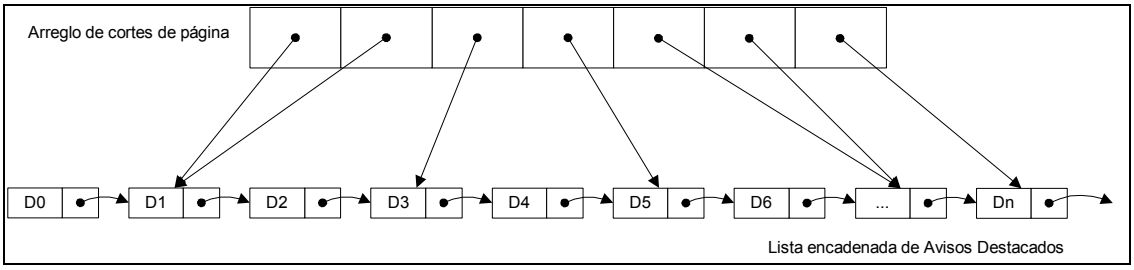

Figura 9: Estructura de Cortes de Páginas

Además de esto, la *solución* almacena la cantidad de páginas procesadas, la cantidad de *destacados fijos* y *flotantes* involucrados en la *ventana* actual, un puntero al primer *aviso lineal* de cada página, el valor de la *función evaluación*, y datos estadísticos de superficie de relleno por página. Como estructuras auxiliares, almacena también un arreglo con el desperdicio por página (distinguiendo *fillers* y *burbujas*), y otros arreglos con los punteros y posiciones de los *avisos destacados* de la *ventana* (uno para los *fijos* y el otro para los *flotantes*), las *secciones* y los *fillers fijos*.

***Lista tabú*:** Fue implementada como un arreglo de longitud D, siendo D la cantidad de *avisos destacados* que intervienen en la *ventana*. En este vector, almacenamos el número de ciclo para el cual la ubicación de un *corte de página* en esa posición deja de ser *tabú*.

***Movimientos*:** Es la estructura que representa el *movimiento* de un *corte de página*, cuando pasa de un lugar a otro. Una colección de estos elementos es utilizada para la generación del *vecindario*. Este es representado con los distintos *movimientos* posibles, a realizar desde una *solución* dada.

***Estructura de comunicación entre ventanas*:** Es una estructura de uso interno que pasa información de una *ventana* a la siguiente. La misma almacena cuál fue la última página procesada, y cuales fueron los últimos avisos (*destacados* y *lineal*) ubicados y su respectiva sección.

5.1.3 Algoritmo principal

A continuación se presenta el pseudocódigo del cuerpo principal del algoritmo:

```
Pagination()
  Load Parameters
  Load Sections
  Load Fixed Fillers
  While exists data
  │ Init Window
  │ Load Displads & Lineads (for this Window)
  │ Generate InitialSolution (for this Window)
  │ BestSolution = TabuSearch(InitialSolution, Window)
  │ Save BestSolution
  EndWhile
End
```

Aquí se puede apreciar básicamente cómo se particionan los datos en *ventanas de optimización*, para su procesamiento. La *Búsqueda Tabú* y la lectura de los avisos, se realizan en base a la cantidad de páginas de la *ventana*.

5.1.4 Búsqueda Tabú

La *Búsqueda Tabú* se realiza de la siguiente manera:

```
TabuSearch(IniSol, Window)
  Declare BestSol, BestLocalSol, ActualSol, TmpSol
  BestSol=IniSol
  ActualSol=IniSol
  BestValue=GetValue(BestSol)
  CycleCounter=0
  While not StopCriteria(BestValue, CycleCounter)
  | BestLocalValue=MAXVALUE
  | For each Mv of Neighborhood(ActualSol)
  | | TmpSol=Move(ActualSol, Mv)
  | | Decoding(TmpSol, Window)
  | | TmpValue=Eval(TmpSol)
  | | If not Tabu(Mv, CycleCounter) or
  | | |   AspCriteria(TmpValue, BestValue)
  | | | If TmpValue < BestLocalValue
  | | | | BestLocalValue=TmpValue
  | | | | BestLocalSol=TmpSol
  | | | | BestLocalMv=Mv
  | | | EndIf
  | | EndIf
  | EndFor
  | If BestLocalValue < BestValue
  | | BestValue= BestLocalValue
  | | BestSol=BestLocalSol
  | EndIf
  | SetTabu(ActualSol, BestLocalMv, CycleCounter)
  | ActualSol=BestLocalSol
  | CycleCounter=CycleCounter+1
  EndWhile
Return BestSol
```

Como se puede apreciar en el pseudocódigo, la estructura del algoritmo se corresponde con el esquema general de *Búsqueda Tabú*, excepto por el llamado a la función *Decoding*. Como hemos explicado antes, la búsqueda se realiza en términos de *movimientos* de los *cortes de página*. Sin embargo, para poder evaluar los resultados de la *solución candidata* (TmpSol), se necesita previamente haber distribuido los *avisos destacados* y *lineales* en las páginas de la *ventana*. Justamente, eso es lo que hace la función *Decoding*, y es descripto en más detalle a continuación.

5.1.5 Ubicación de avisos destacados y de relleno en la página

Como acabamos de explicar, la función *Decoding* se encarga de la ubicación de los distintos tipos de avisos en la página. La misma se realiza en la siguiente secuencia:

1. Se ubican los *avisos destacados* "*fijos*", que tengan su **página** y **posición** pre-asignada por el usuario.

2. Se ubican los *avisos destacados* "*fijos*", que tengan sólo su **posición** pre-asignada por el usuario, y que hayan sido asignados por el algoritmo de *Búsqueda Tabú* a la **página** en cuestión.
3. Se ubican los *avisos de relleno* "*fijos*", que tengan su **página** y **posición** pre-asignada por el usuario (como *espacio de reserva*).
4. Se ubican los *avisos* "*flotantes*" (*destacados* y de *relleno*), que tengan sólo su **página** pre-asignada por el usuario, junto con los *destacados* que hayan sido asignados por el algoritmo de *Búsqueda Tabú* a la **página** en cuestión. Es precisamente en este punto, donde intervienen las diferentes *heurísticas* desarrolladas (de acuerdo al estilo de *layout* elegido).
5. Finalmente, de ser necesario, se rellenan los agujeros generados (como *burbujas*).

5.1.6 Ubicación de avisos lineales (y fillers)

En una segunda etapa de la función *Decoding*, los *avisos lineales* se "fluyen" columna a columna en el espacio remanente, por encima de los avisos previamente ubicados. Este paso incluye la ubicación de los *avisos lineales* y los *encabezados de sección*. En el caso de ser necesario, se recurre al relleno con *fillers* para evitar cortes de columna "indeseados". Esto es, por ejemplo, en el caso de ocurrir un corte de columna inmediatamente después de un *encabezado de sección*. O bien, cuando no quepa el siguiente *aviso lineal* a ubicar en el espacio disponible en la columna actual.

5.1.7 Función Evaluación

La implementación de la función evaluación aplica todos los criterios previamente descriptos. La violación de estos criterios, según el caso, es penalizada de distinta manera. Hay características que no pueden dejar de cumplirse, como por ejemplo, que no haya avisos en *Overflow*. De producirse alguna condición de este tipo, la pena aplicada debería ser lo suficientemente alta como para que el algoritmo por sí solo, considere inviable esa solución. Con este objetivo, algunas de las características tienen una doble penalización: una *absoluta* (en general muy alta), que castiga la simple existencia de la condición, y otra relativa (usualmente más baja), que evalúa "en qué medida" se ha producido. Esta última clase de penalización podrá ser proporcional a la superficie, la cantidad de avisos o la distancia, según el caso. El objetivo de esta segunda clase de penalización es que el algoritmo tenga la forma de ponderar y relativizar soluciones defectuosas, para poder orientar la búsqueda hacia soluciones de mejor calidad.

La evaluación de los distintos criterios se hace de la siguiente manera:

- ***Overflow***: Esta es una de las condiciones que no son aceptables en una *solución final*. Por esta razón, tiene la doble penalización mencionada anteriormente. Se compone entonces de un valor fijo (muy alto) que penaliza la existencia, y de otro variable (más pequeño), proporcional al área de los avisos que no caben en la página.
- ***Avisos fuera de orden***: Dada la forma en que implementamos el algoritmo, no es posible alterar el orden de los *avisos destacados flotantes*, con lo que no nos tenemos que preocupar porque esta condición ocurra.
- ***Distancia al encabezado***: Penalizamos la distancia de cada *destacado*, respecto del inicio de su *sección* (proporcionalmente a la cantidad de páginas). De esta manera, estamos "estimulando" al algoritmo para que ubique a los *avisos destacados* tan cerca del *encabezado* como sea posible (e idealmente, en la misma página).
- ***Aviso previo al encabezado***: Si la distancia mencionada anteriormente es negativa, significa que existen *avisos destacados* en páginas previas al comienzo de la *sección*. Esta condición

no es conveniente. Por este motivo, al igual que en el caso del *Overflow*, se fija un valor absoluto muy alto ante esta condición. Al mismo tiempo, también se penaliza con un valor relativo más bajo por cada aviso que se ubique antes que el *encabezado* (con un costo proporcional a la cantidad de páginas *fuera de sección*).

- ***Superficie de relleno***: Elegimos penalizar en forma distinta la generación de *burbujas* y de *fillers*, ya que esta última es muy común que se produzca al finalizar cada columna, debido a que no se pueden hacer alteraciones en el orden de los *avisos lineales* (por estar estos ordenados alfabéticamente). En cambio, las *burbujas* se pueden producir por una mala disposición de los *avisos destacados*, y es lo que el algoritmo debería tratar de evitar con más énfasis. La lógica entonces indica que se deberían penalizar con valores más altos las *burbujas* que los *fillers*. La penalización de estos objetos, es siempre proporcional a la superficie de los mismos.
- ***Preferencia par o impar***: La violación de esta preferencia está también penalizada doblemente, mediante un valor absoluto y uno relativo (según la superficie del aviso en esta condición). Pero la misma no se realiza tan duramente como los casos anteriores, debido a que serían aceptables soluciones con *avisos destacados* que no respeten dicha preferencia.

5.1.8 Parámetros del módulo

El *Motor* recibe sus parámetros desde la *Interfase Gráfica*, a través del archivo *ts.ini*. Los parámetros que recibe detallan distintas propiedades relativas tanto al formato y estilo de página, como así también, a cuestiones propias de la *Búsqueda Tabú*. A grandes rasgos, los aspectos que se pueden parametrizar son los siguientes:

- Cantidad de filas y columnas de las páginas.
- Forma de agrupar los *avisos destacados* en la página: horizontalmente (“acostando”) o verticalmente (“apilando”).
- Estilo de distribución de los *avisos destacados*: desde el borde exterior de la página, hacia el interior (independientemente, para la página de la izquierda y de la derecha).
- Tamaño de *ventana* y *solapamiento*.
- Cantidad máxima de iteraciones de la *Búsqueda Tabú* (para la ventana).
- Umbral mínimo de la *función evaluación* (como condición adicional de parada).
- Porcentaje de *vecinos* a analizar sobre el total.
- Parámetros relativos a la *retención* en la *lista tabú*.
- Parámetros relativos a la *función evaluación*.

En el punto 5.3.2.2 se detalla el contenido del archivo *ts.ini*.

5.2 Interfase Gráfica

La interfase gráfica es una aplicación desarrollada con Visual Basic 6. Básicamente, es una herramienta que permite visualizar publicaciones mediante la graficación de sus páginas con sus correspondientes elementos: *avisos destacados*, *avisos lineales*, *avisos de relleno* (*fillers* y *burbujas*) y encabezados de sección. También, de forma simple e intuitiva, brinda la posibilidad de modificar ciertas propiedades de los *avisos destacados*, y la creación y modificación de *avisos de relleno*, de esta manera, se le permite al usuario incluir sus propios criterios a la publicación.

La aplicación se basa en tres grandes componentes:

- Módulo principal
- Objeto "doble página"
- Módulo de manejo de archivos

El Módulo Principal es la ventana de la aplicación que incluye el menú de opciones, la barra de herramientas, línea de estado y el espacio en donde se despliegan los objetos "doble página".

El Objeto "doble página" es un módulo de clase que representa gráfica y funcionalmente las dos páginas abiertas de una publicación. Y es quien controla todas las operaciones de los elementos de la publicación.

El Módulo de manejo de archivos es una librería de procedimientos que permite operar sobre los archivos de la publicación. Por ej., al mover un *aviso destacado*, los archivos que contienen el elemento en cuestión deben actualizarse para reflejar el nuevo estado.

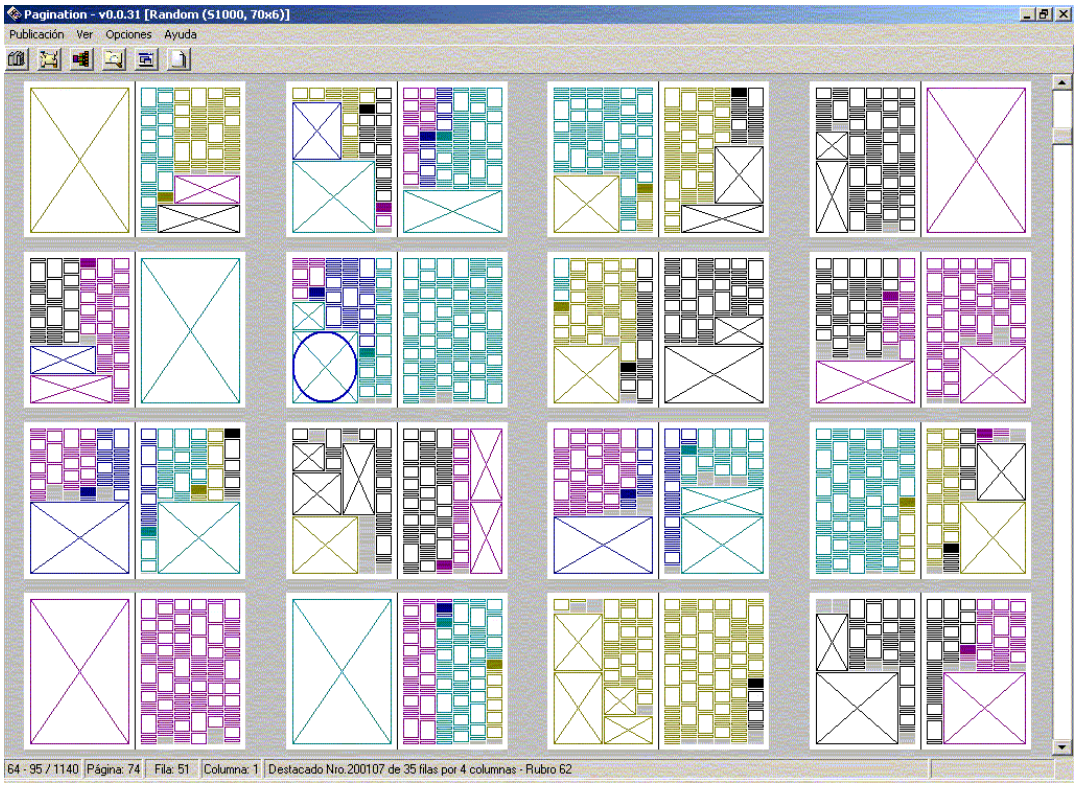

Figura 10: Interfase Gráfica del sistema

5.2.1 Funcionalidad

A continuación enumeramos las principales funciones de la Interfase Gráfica:

- Visualización de una publicación: navegación, búsqueda de páginas, composición, identificación de elementos
- Repaginación de la publicación por fracción de páginas o por el total de la publicación.
- Reserva de espacio mediante la creación de *avisos de relleno*
- Parametrizaciones: estilo de páginas, *Búsqueda Tabú*, función evaluación y layout de páginas.
- Modificaciones del usuario: fijación de *avisos destacados* y *fillers* en determinada posición y/o página, y preferencia de páginas par o impar.

En el Apéndice C presentamos una descripción detallada de esta funcionalidad y la forma de operarla.

5.2.2 Representación gráfica de los elementos

La representación gráfica de los elementos de una publicación son los siguientes:

- *Avisos destacados*: cajas marcadas con una cruz

- *Avisos lineales*: cajas vacías
- *Avisos de relleno*: cajas rellenas de color gris
- Encabezados de sección: cajas rellenas

Las propiedades de los elementos *destacados* y de relleno se denotan según su color:

- Elementos con posición fija: cruz de color roja
- Elementos con página fija: caja de color roja

Existen dos modos de visualización: El “modo color” (modo *default*), distingue los elementos de una misma sección asignando distintos colores para distintas secciones. El otro, no hace distinción según la sección a la que pertenecen, graficándolos a todos con color negro.

Una función útil para el usuario es realizar un click sobre un elemento para determinar visualmente todos los otros elementos que comparten la misma sección. Todos estos se "iluminan" con color azul.

Una página remarcada con color rojo indica que es una página con *overflow*. Esto significa que la misma contiene elementos (*avisos destacados*, *avisos lineales* y *avisos de relleno*) que no pudieron ser ubicados en la página por falta de espacio.

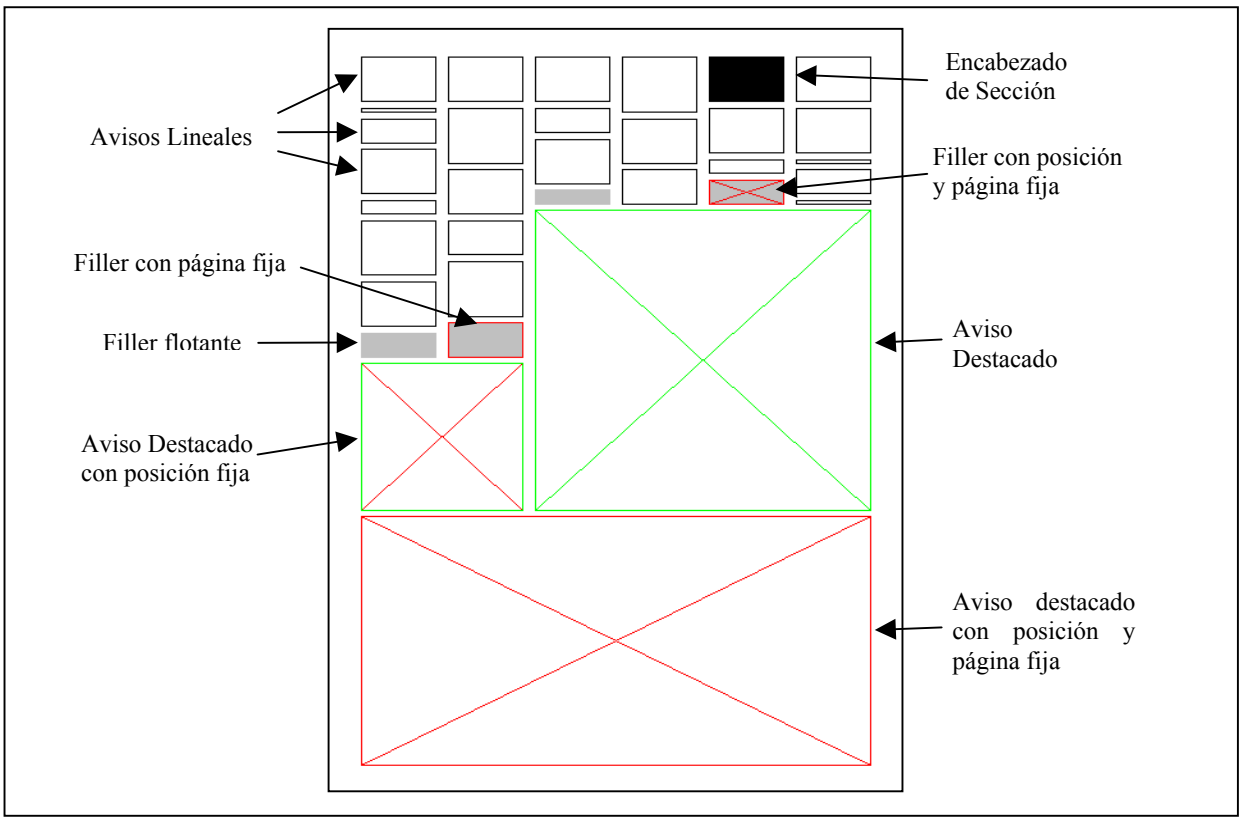

Figura 11: Representación gráfica de los elementos de una publicación

5.3 Comunicación entre módulos

Los dos módulos descriptos anteriormente se comunican mediante un conjunto de archivos. Los mas importantes son:

Archivos de tipo .IN: son archivos que contienen los elementos de la publicación ordenados por rubros o secciones. El nombre del archivo indica el número de la sección y todos los elementos incluidos pertenecen a esta. De estos archivos, el Motor toma sus datos de entrada.

Archivos de tipo .PAG: son archivos que contienen los elementos de la publicación ordenados por página. El nombre del archivo indica el número de la misma y todos los elementos de la página son incluidos aquí. A través de este tipo de archivos, el Motor devuelve el resultado del proceso a la Interfase Gráfica.

Esta separación fue realizada debido a que, por cuestiones de diseño, el Motor necesita sus datos ordenados por sección, mientras que la Interfase Gráfica los usa por página.

Los archivos tienen formato ASCII y contienen sus datos agrupados por tipo de aviso. Así pues, tendremos un conjunto de registros para *avisos destacados*, otros conjuntos para *avisos lineales*, etc.

Además, existen archivos de tipo *.INI,* los cuales contienen los valores de los parámetros utilizados por el Motor y la Interfase Gráfica.

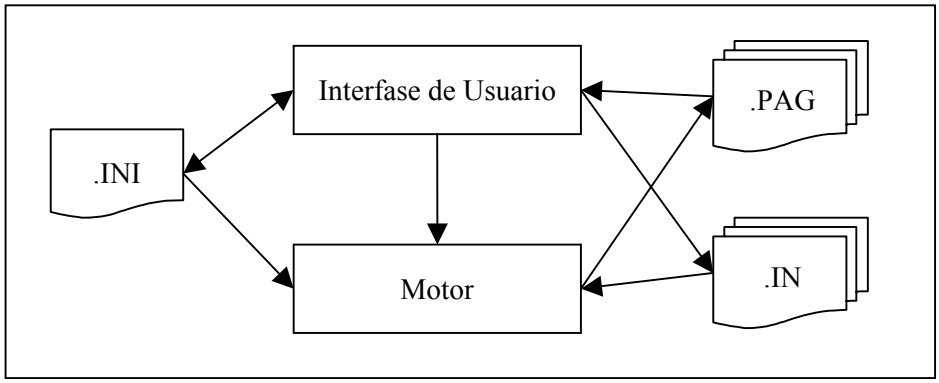

Figura 12: Módulos del sistema y sus interrelaciones

5.3.1 Archivos de entrada a la Interfase Gráfica

5.3.1.1 Archivo *Pagination.ini*

En el archivo *pagination.ini* se almacenan todos los parámetros de una publicación. Los mismos se encuentran ubicados en las carpetas correspondientes a cada una de ellas. Contiene seis grupos de parámetros:

- *Páginas*: Características de las páginas. Sirven para la visualización en pantalla.
- *Parámetros*: Atributos de la publicación (filas, columnas, páginas).
- *Gutter*: Espacios entre avisos. Sirven para la visualización en pantalla.
- *Fechas*: Fecha de creación y ultima modificación de la publicación
- *TS*: Parámetros enviados al Motor para la paginación de la publicación.
- *Layout*: Parámetros enviados al Motor para la composición de las páginas.

A continuación, vemos un archivo de ejemplo, con una breve descripción de cada uno de los parámetros:

Pagination.ini:

```
[Pagina]
Tipo=3                          Indica el tipo de página (A3, A4, Legal, etc).
```

Alto=297	Alto de la página en milímetros.
Ancho=210	Ancho de la página en milímetros.
Orientacion=0	0: vertical, 1: apaisada.
Margen Superior=10	Margen superior en milímetros.
Margen Inferior=10	Margen inferior en milímetros.
Margen Externo=10	Margen externo (de la doble página) en milímetros.
Margen Interno=10	Margen interno (de la doble página) en milímetros.
[Parametros]	
filas=70	Cantidad de filas por página (en módulos).
columnas=6	Cantidad de columnas por página.
paginas=51	Páginas de la publicación.
[Gutter]	
Alto=2	Separación vertical entre avisos.
Ancho=2	Separación horizontal entre avisos.
[Fechas]	
Creacion=11/11/01 12:33	Fecha y hora de creación de la publicación.
Modificacion=25/11/01 08:26	Fecha y hora de última modificación de la publicación.
[TS]	
WSIZE=10	Tamaño, en páginas, de la ventana de procesamiento.
WSSIZE=1	Tamaño, en páginas, del solapamiento entre ventanas.
MAXCYCLES=1000	Cantidad máxima de ciclos de la *Búsqueda Tabú*.
MAXNEIGH=40	Porcentaje a procesar sobre el total de los vecinos.
RETENTION_A=200	Valor cota inferior en el cálculo de la retención. (a $\sqrt{d}$ /100).
RETENTION_B=300	Valor cota superior en el cálculo de la retención. (b $\sqrt{d}$ /100).
MINVALSOLUTION=1000	Valor mínimo de la solución usado como criterio de parada.
OVERFLOW_AREA=500	Penaliz. del *Overflow* (por área).
OVERFLOW=50000	Penaliz. del *Overflow* (valor fijo).
BEFORE_SECTION_AREA=800	Penaliz. de avisos fuera de sección (por área).
BEFORE_SECTION=75000	Penaliz. de avisos fuera de sección (valor fijo).
PREFER_EVEN_ODD_AREA=2	Penaliz. de violación preferencia par/impar (por área).
PREFER_EVEN_ODD=100	Penaliz. de violación preferencia par/impar (valor fijo).
BUBBLE_AREA=8	Penaliz. de la existencia de *burbujas* (por área).
FILLER_AREA=2	Penaliz. de la existencia de *fillers* (por área).
DIST_FROM_SECT_START=100	Penaliz. Distancia Encab-Dest (por aviso y # páginas).
[LAYOUT]	
MIN_FILL_BELOW_HEADER=2	Espacio generado entre encabezados para rubros sin *lineales*.
RPAGE_FROM_EXTERIOR=1	Avisos hacia el exterior (página derecha).
LPAGE_FROM_EXTERIOR=1	Avisos hacia el exterior (página izquierda).
PAGE_SMALLEST_DEPTH=1	0: “Apilar”, 1: “Acostar”.

5.3.1.2 Archivos de tipo .PAG

Para facilitar la navegabilidad en forma aleatoria entre las páginas de una publicación, se diseño el repositorio de forma tal que la información esté agrupada y ordenada por página. La publicación consta entonces, de un conjunto de archivos, uno por página, almacenando todos los elementos que le corresponden. A continuación detallamos cada uno de los elementos que componen estos archivos:

- **Headers:**
  - Id: Identificador de sección.

  - Filas: Tamaño del encabezado de sección (en módulos).
  - Fila: Fila en la que fue ubicado.
  - Columna: Columna en la que fue ubicado.
- **DisplAds:**
  - Id: Identificador del *aviso destacado*.
  - Filas: Tamaño del *aviso destacado* (en módulos).
  - Columnas: Tamaño del *aviso destacado* (en columnas).
  - *Aviso Lineal* relacionado: Identificador del *aviso lineal* asociado (si existe).
  - Sección: Identificador de la sección correspondiente.
  - Fila: Fila en la que fue ubicado.
  - Columna: Columna en la que fue ubicado.
  - Preferencia página: De tener preferencia asignada, si es página par o impar.
  - Predeterminado: De tener posición preasignada, si es la página, la posición, o ambas.
- **LineAds:**
  - Id: Identificador de *aviso lineal*.
  - Filas: Tamaño del *aviso lineal* (en módulos).
  - *Aviso Destacado* relacionado: Identificador del *aviso destacado* asociado (si existe).
  - Sección: Identificador de la sección correspondiente.
  - Fila: Fila en la que fue ubicado.
  - Columna: Columna en la que fue ubicado.
- **Fillers:**
  - Id: Identificador del aviso de relleno predefinido.
  - Filas: Tamaño del *filler* (en módulos).
  - Columnas: Tamaño del *filler* (en columnas).
  - Fila: Fila en la que fue ubicado.
  - Columna: Columna en la que fue ubicado.
  - Predeterminado: Si la posición preasignada es de página, de posición, o de ambas.

Archivo de ejemplo:

```
Headers
<section_id> <sz_row> <row> <col>
0028  2     20    1
0029  3     25    1
.
.
.
End
DisplAds
<display_id> <sz_row> <sz_col> <rel_linead_id> <section_id> <row> <col>
<page_pref> <predet>
200061      18    3     -     0025  54    0     -     0
200062      12    2     -     0025  60    3     -     0
200063      12    2     -     0030  48    3     -     0
.
.
.
End


LineAds
<linead_id> <sz_row> <rel_displad> <sect_id> <row> <col>
301005      1     -     0027  0     0
301006      1     -     0027  1     0
301007      2     -     0027  2     0
301008      10    -     0027  4     0
.
.
```

```
.
End
Fillers
<filler_id> <sz_row> <sz_col> <row> <col> <predet>
502200      4     1     50    0     0
502201      2     1     52    1     0
502202      1     1     53    2     0
.
.
.
End
```

5.3.2 Archivos de entrada al Motor

5.3.2.1 Archivos de tipo .IN

Es un conjunto de archivos que registran todos los elementos de la publicación, ordenados por rubro o sección. Los usa el Motor como archivos de entrada de datos. Existen tres tipos de archivos dentro de esta estructura:

- Sections.in
- SectXXXX.in
- Fillers.in

Archivo *Sections.in*

Contiene todas las propiedades de cada una de las secciones de la publicación. El formato del registro es el siguiente:

- Id: Identificador de rubro.
- Cantidad de DisplAds: Cantidad de *avisos destacados* en la sección.
- Area de DisplAds: Area total de los *avisos destacados* de la sección.
- Cantidad de LineAds: Cantidad de *avisos lineales* en la sección.
- Area de LineAds: Area total de los *avisos lineales* en la sección.
- Tamaño del Header: Tamaño del *encabezado de sección* (en módulos).
- Página del Header: Página en que fue ubicado el *encabezado de sección*.
- Fila del Header: Fila en la que fue ubicado el *encabezado de sección*.
- Columna del Header: Columna en la que fue ubicado el *encabezado de sección*.
- Pág. primer *destacado*: Página en la que se ubica el primer *destacado* de la sección.
- Pág. último *destacado*: Página en la que se ubica el último *destacado* de la sección.
- Pág. último *lineal*: Página en la que se ubica el último *aviso lineal* de la sección.

Archivo de ejemplo:

```
Sections:
<section_id> <displads_nr> <displads_area> <lineads_nr> <lineads_area>
<header_sz_row> <header_page> <header_row> <header_col>
<fist_displad_page>  <last_displad_page> <last_linead_page>
0001  1     105   15    38    5     1     0     0     1     1     1
0002  4     385   151   447   5     1     6     1     1     3     3
0003  3     287   107   318   5     3     20    2     3     5     4
0004  1     420   14    40    5     4     10    4     6     6     4
0005  0     0     2     7     5     4     55    4     -1    -1    4
```

```
0006  3     322   161   479   5     4     0     5     7     8     7
0007  2     196   87    254   5     7     0     2     8     8     9
0008  1     210   13    36    5     9     15    3     9     9     9
0009  0     0     2     2     5     9     21    4     -1    -1    9
0010  6     336   161   443   5     9     28    4     10    10    13
.
.
.
End
```

Archivos *Sect<Nro. de Sección>.in*

Contiene todos los elementos (*avisos destacados* y *avisos lineales*) de la correspondiente sección. El formato del mismo es:

- **DisplAds:**
  - Id: Identificador del *aviso destacado*.
  - Filas: Tamaño del *aviso destacado* (en módulos).
  - Columnas: Tamaño del *aviso destacado* (en columnas).
  - *Aviso Lineal* relacionado: Identificador del *aviso lineal* asociado (si existe).
  - Página: Página en la que fue ubicado.
  - Fila: Fila en la que fue ubicado.
  - Columna: Columna en la que fue ubicado.
  - Preferencia página: De tener preferencia asignada, si es página par o impar.
  - Predeterminado: De tener posición preasignada, si es la página, la posición, o ambas.
- **LineAds:**
  - Id: Identificador de *aviso lineal*.
  - Filas: Tamaño del *aviso lineal* (en módulos).
  - *Aviso Destacado* relacionado: Identificador del *aviso destacado* asociado (si existe).
  - Página: Página en la que fue ubicado.
  - Fila: Fila en la que fue ubicado.
  - Columna: Columna en la que fue ubicado.

Ejemplo de archivo:

```
DisplAds:
<display_id> <sz_row> <sz_col> <rel_linead_id> <page> <row> <col> <page_pref>
<predet>
200167  35    6      -      97     35     0      -      0
200168  14    5      -      97     21     1      -      0
200169  14    3      -      97     7      3      -      0
.
.
End
```

```
LineAds:
<linead_id>  <sz_row> <rel_displad_id> <page> <row> <col>
307007  2     -      96     15     5
307008  1     -      96     17     5
307009  6     -      96     18     5
307010  1     -      96     24     5
307011  2     -      96     25     5
307012  2     -      96     27     5
.
.
End
```

Archivo *Fillers.in*

Contiene los todos los *avisos de relleno* prefijados de una publicación. El formato de los registros es el siguiente**:**

- Id: Identificador del *aviso de relleno* predefinido.
- Filas: Tamaño del *aviso de relleno* (en módulos).
- Columnas: Tamaño del *aviso de relleno* (en columnas).
- Página: Página en la que fue ubicado.
- Fila: Fila en la que fue ubicado.
- Columna: Columna en la que fue ubicado.
- Predeterminado: Si la posición preasignada es de página, de posición, o de ambas.

Archivo de ejemplo:

```
Fillers:
<filler_id> <sz_row> <sz_col> <page> <row> <col> <predet>
520001      70    6     1     0     0     7
500004      2     1     2     40    5     0
.
.
.
End
```

5.3.2.2 Archivo *Ts.ini*

El archivo *ts.ini* es generado por la Interfase Gráfica como primer paso en el proceso de paginación. Su propósito es realizar el pasaje de parámetros hacia el Motor.

Archivo de ejemplo:

INIPAGE 1	Página a partir de la cual se comienza a paginar.
NPAGES 4	Número de páginas a generar a partir de la página inicial.
COLS 6	Columnas por página.
ROWS 70	Filas por página.
WSIZE 10	Tamaño, en páginas, de la ventana de procesamiento.
WSSIZE 1	Tamaño, en páginas, del solapamiento entre ventanas.
MAXCYCLES 1000	Cantidad máxima de ciclos de la *Búsqueda Tabú*.
MAXNEIGH 40	Porcentaje a procesar sobre el total de los vecinos.
RETENTION_A 200	Valor cota inferior en el cálculo de la retención. (a $\sqrt{d}$ /100).
RETENTION_B 300	Valor cota superior en el cálculo de la retención. (b $\sqrt{d}$ /100).
MINVALSOLUTION 0	Valor mínimo de la solución usado como criterio de parada.
OVERFLOW_AREA 500	Penaliz. del *Overflow* (por área).
OVERFLOW 50000	Penaliz. del *Overflow* (valor fijo).
BEFORE_SECTION_AREA 800	Penaliz. de avisos fuera de sección (por área).

BEFORE_SECTION 75000	Penaliz. de avisos fuera de sección (valor fijo).
PREFER_EVEN_ODD_AREA 2	Penaliz. de violación preferencia par/impar (por área).
PREFER_EVEN_ODD 100	Penaliz. de violación preferencia par/impar (valor fijo).
BUBBLE_AREA 8	Penaliz. de la existencia de *burbujas* (por área).
FILLER_AREA 2	Penaliz. de la existencia de *fillers* (por área).
DIST_FROM_SECT_START 100	Penaliz. Distancia Encab-Dest (por aviso y # páginas).
MIN_FILL_BELOW_HEADER 2	Espacio generado entre encabezados para rubros sin *lineales*.
RPAGE_FROM_EXTERIOR 1	Avisos hacia el exterior (página derecha).
LPAGE_FROM_EXTERIOR 1	Avisos hacia el exterior (página izquierda).
RPAGE_SMALLEST_DEPTH 1	0: “Apilar”, 1: “Acostar” (página derecha).
LPAGE_SMALLEST_DEPTH 1	0: “Apilar”, 1: “Acostar” (página izquierda).

5.4 Requerimientos de hardware

Si bien el sistema puede correr en cualquier PC típica con sistema operativo Windows 95 o superior, se recomienda que la misma posea la siguiente configuración mínima:

- Procesador: Pentium II o superior
- Memoria RAM: 64Mb.
- Espacio disponible en disco: 10Mb.
- Monitor: 17" color
- Mouse

6 RESULTADOS

En este capítulo exponemos los resultados obtenidos en las corridas realizadas con diferentes lotes de prueba. Incluimos un detalle de dichos lotes, utilizados para el ajuste de parámetros del algoritmo implementado. Analizamos el desempeño del prototipo desarrollado, tanto desde el punto de vista del algoritmo de *Búsqueda Tabú* implementado, como así también respecto de los criterios gráficos considerados.

6.1 Lotes de prueba

Dada la dificultad de contar con datos reales contra los cuales comparar el comportamiento del prototipo implementado, decidimos realizar las pruebas y ajustes del algoritmo con algunos lotes semi-reales y otros aleatorios. Esto de debe, principalmente, a que dicha información es propiedad de las empresas telefónicas y editoriales que los administran.

6.1.1 Las Páginas Amarillas

Generamos un lote de prueba semi-real de 51 páginas, extraído de una publicación de la guía de *Las Páginas Amarillas* de Capital Federal [PA99]. Como se tornaba prácticamente imposible relevar y medir todos los avisos de la extracción realizada, tomamos la decisión de volcar de manera exacta sólo la información relativa a los *avisos destacados* de cada sección. En cuanto a los *avisos lineales*, tomamos la superficie abarcada para cada rubro, y generamos aleatoriamente avisos de diferentes dimensiones, por áreas equivalentes. Como resultado, obtuvimos una "publicación tipo", de características muy similares a la publicación real. La misma está formada por una grilla de página de 72 filas por 5 columnas, determinando 360 módulos (o unidades) por página.

El hecho de realizar pruebas con este lote semi-real nos permitió además ajustar los valores de la función evaluación para aproximar el estilo de paginación y layout a un ambiente real.

Además, decidimos generar un segundo lote de prueba basado en el anterior, pero fijando un aviso en una página en particular. Esto lo hicimos con la intención de verificar la variación de los resultados obtenidos ante la intervención de un operador. Al mismo tiempo, este cambio generaría resultados más parecidos a la publicación comercial.

En adelante, el primer lote de prueba será referenciado como **PASF**, mientras que el segundo, como **PACF**. Las características de estos lotes se detalla en la siguiente tabla:

Elementos	Cantidad	Area	% Area	Prom. Area/Elem
Avisos Destacados	126	12.036	66,72	95,52
Avisos Lineales	2.280	5.799	32,15	2,54
Encabezados de Sección	75	203	1,13	2,7
TOTAL	**2.481**	**18.038**	**100,00**	

Tabla 1: Características del lote de prueba PASF/PACF

6.1.2 Aleatorios

Armamos dos publicaciones aleatoriamente, mediante un generador random que implementamos. La finalidad fué probar y analizar el comportamiento del prototipo desarrollado con publicaciones de diferentes características. Sobre todo, respecto de la cantidad de avisos que las conforman.

Las publicaciones generadas tienen una grilla de página de 70 filas por 6 columnas, determinando 420 módulos (o unidades) por página.

Tanto la cantidad como las dimensiones de los avisos fueron generadas en forma aleatoria. Las proporciones entre superficie de *lineales* y *destacados* fueron fijadas con distintos valores para cada uno de los lotes: 50% de *avisos lineales* en uno, y 60% en el otro. Otra diferencia fundamental entre los lotes, es el volumen. El primero, al que llamaremos **RND100** (o eventualmente **RND**), contiene 100 secciones; mientras que el segundo lote, al que llamaremos RND1000, contiene 1.000 secciones. En las siguientes dos tablas se muestran las características de estos lotes:

Elementos	Cantidad	Area	% Area	Prom. Area/Elem
Avisos Destacados	202	21.938	47,20	108,60
Avisos Lineales	8.341	24.044	51,73	2,88
Encabezados de Sección	100	500	1,07	5,00
TOTAL	**8.643**	**46.482**	**100,00**	

Tabla 2: Características del lote de prueba RND100

Elementos	Cantidad	Area	% Area	Prom. Area/Elem
Avisos Destacados	1.554	179.508	38,70	115,51
Avisos Lineales	98.619	279.370	60,22	2,83
Encabezados de Sección	1.000	5.000	1,08	5,00
TOTAL	**101.173**	**463.878**	**100,00**	

Tabla 3: Características del lote de prueba RND1000

6.2 Ajuste de parámetros

El ajuste de parámetros fue realizado uno a uno y analizados forma independiente. Al ir avanzando con el proceso, los mejores valores encontrados fueron fijados para las pruebas posteriores.

Los parámetros iniciales fueron establecidos en base a las pruebas preliminares realizadas.

Para determinar la cantidad de iteraciones a realizar en el ajuste de los parámetros, hicimos 10 corridas de 10 mil iteraciones para cada uno de los lotes de prueba. Como en los resultados preliminares obtenidos observamos que la convergencia siempre se producía para no más de 1.000 iteraciones, decidimos acotar a dicho valor la cantidad de ciclos para todas las demás pruebas realizadas.

El proceso comenzó con el ajuste de los parámetros referentes a la Función Evaluación, y continuó con los parámetros de la *Búsqueda Tabú*. En el primero de los casos, y con el único propósito de acotar la cantidad de pruebas a realizar, decidimos no considerar la penalización por violación de página par/impar.

La partición en ventanas podía complicar el análisis de los resultados, ya que por el solapamiento de las mismas, los valores acumulados de las soluciones parciales no son el fiel reflejo de los valores reales. Por este motivo, hemos decidido tomar las mediciones en base a una sola ventana de procesamiento. El tamaño de ventana seleccionado fue de 20 páginas.

Para cada ajuste de parámetro realizamos 10 corridas por cada uno de los lotes, generando de esta manera, una gran cantidad de datos para analizar.

6.2.1 Ajuste de parámetros para la Función Evaluación

Overflow:

El parámetro que ajustamos es el que corresponde a la penalización variable ante esta condición. El valor asignado se multiplica por la superficie de los *avisos destacados* en *Overflow*. En este sentido, en caso de existir *overflow* para alguna página, sumamos además un valor constante lo suficientemente elevado como para "descalificar" estas soluciones indeseables. Este valor fue, arbitrariamente, fijado en 50.000 para todas las pruebas realizadas.

Para las corridas realizadas, estos fueron los valores obtenidos:

Overflow	PACF	PASF	RND
0	10.333,10	10.610,70	5.577,90
100	41.473,90	68.060,60	11.904,60
200	30.471,00	38.349,10	5.321,00
300	30.445,00	49.622,60	5.321,00
400	30.465,00	51.000,00	5.321,00
500	30.511,00	38.263,20	5.321,00
600	36.670,50	43.494,40	5.321,00
700	30.507,00	44.785,00	5.321,00
800	30.539,00	37.648,10	5.321,00
900	30.467,00	67.909,10	5.321,00
1000	30.649,80	55.829,90	5.321,00

Tabla 4: Comparativa Func. Eval. – Overflow

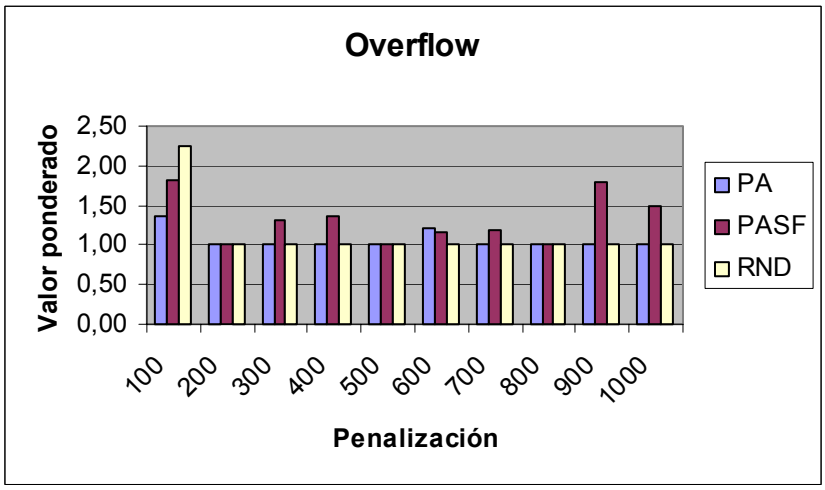

Figura 13: Comparativa Func. Eval. – Overflow

Aquí vemos que, salvo para valores muy pequeños asignados a esta penalización, este parámetro no afecta el buen comportamiento para los lotes **PACF** y **RND**.

Para el caso específico del lote **PASF**, se ve claramente que no es una buena medida penalizarlo con valores muy bajos o muy altos (100 en un extremo, y 900 y 1000 en el otro). Para valores intermedios, si bien en algunos casos no llegó a la solución óptima, no se han producido anormalidades "graves". Es decir, todos los *avisos destacados* entraron en las páginas asignadas. El valor de 500 puede ser una buena medida, ya que además de haber producido las mejores soluciones en los tres casos, es un valor intermedio entre los extremos marcados por el lote **PASF**.

Burbujas:

Este parámetro ajusta la penalización por superficie, ante la existencia de *burbujas*. Recordemos que las *burbujas* son aquellos desperdicios generados exclusivamente entre los *avisos destacados*, con lo que la frecuencia de ocurrencia de estos elementos no es muy alta.

Las pruebas realizadas arrojaron los siguientes valores:

Burbujas	PACF	PASF	RND
1	30.449,00	37.499,00	5.105,40
2	30.363,00	38.737,30	5.105,40
3	36.491,30	43.232,00	5.164,20
4	30.357,00	45.801,90	5.181,00
5	30.441,00	49.677,00	5.181,00
6	30.331,00	38.796,30	5.181,00
7	30.369,00	55.941,00	5.181,00
8	30.383,00	32.766,50	5.181,00
9	30.343,00	54.918,20	5.181,00
10	30.351,00	56.402,20	5.181,00

Tabla 5: Comparativa Func. Eval. – Burbujas

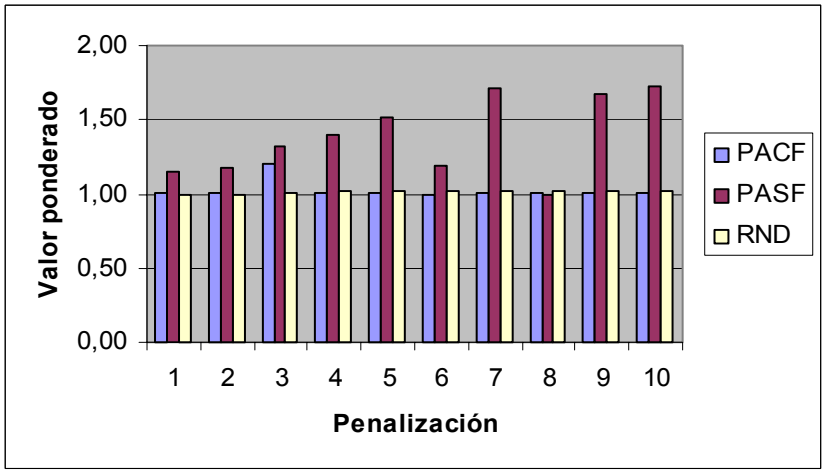

Figura 14: Comparativa Func. Eval. – Burbujas

Los valores que podemos observar, nos indican que si bien el valor de penalización 8 ha dado los mejores resultados en todos los casos, este parámetro no tiene demasiada influencia en el comportamiento del algoritmo. Los lotes **PACF** y **RND**, salvo en un caso, no han variado su comportamiento habiendo modificado este valor. Para el lote **PASF**, si bien el mejor valor de evaluación es alcanzado para una penalización de 8, todos los valores son razonablemente buenos, considerando que no se producen anormalidades.

Fillers:

Con este valor ajustamos la penalización asignada a los rellenos generados al pie de las columnas. A diferencia de las *burbujas*, este elemento se genera muy frecuentemente, y en consecuencia, su valor de penalización debería ser menor que el anterior.

Las pruebas realizadas arrojaron los siguientes resultados:

Fillers	PACF	PASF	RND
1	30.245,00	57.855,10	4.819,00
2	30.323,00	48.947,20	4.819,00
3	30.307,00	49.647,70	4.819,00
4	30.323,00	49.858,00	4.819,00
5	30.269,00	62.329,00	4.819,00
6	42.470,30	50.451,80	4.819,00
7	36.374,70	44.246,30	5.008,00
8	42.496,40	49.678,20	4.987,00
9	30.265,00	49.051,20	5.029,00
10	30.261,00	45.430,70	5.029,00

Tabla 6: Comparativa Func. Eval. – Fillers

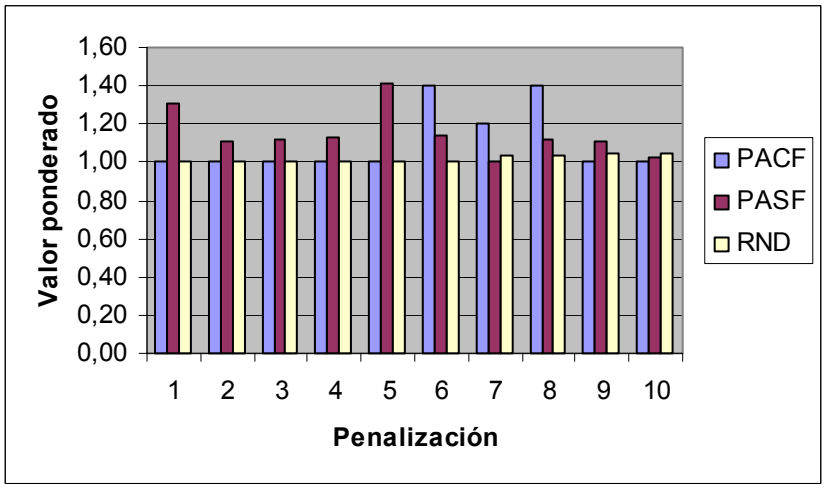

Figura 15: Comparativa Func. Eval. – Fillers

Las observaciones que podemos hacer en este caso son muy subjetivas. Los valores intermedios parecen ser los que peores resultados dan, mientras que valores de penalización 2, 3 o 4, o en el otro extremo, 9 y 10, generan un comportamiento razonablemente bueno. Por lo expresado anteriormente, en cuanto a que la penalización de *fillers* debe ser menor que la de las *burbujas*, seleccionamos el valor de penalización 2 como la mejor opción.

Avisos Fuera de Sección:

El ajuste en cuestión, penaliza por cantidad de elementos, la presencia de *avisos destacados* que precedan a la ubicación del *encabezado de sección* correspondiente. En este sentido, en caso de existir algún *aviso destacado* en esta condición, sumamos además un valor constante lo suficientemente elevado como para "descalificar" esta clase de soluciones. Este valor fue, arbitrariamente, fijado en 75.000 para todas las pruebas realizadas.

Los valores obtenidos de las pruebas son:

Fuera de sección	PACF	PASF	RND
0	78.507,00	75.541,40	75.347,80

100	82.425,00	82.168,70	77.074,00
200	85.883,60	80.071,80	48.798,80
300	65.027,20	55.687,50	12.461,80
400	30.345,00	49.146,80	5.070,00
500	48.546,00	50.230,00	5.070,00
600	30.305,00	32.865,40	5.070,00
700	30.305,00	45.853,60	5.070,00
800	30.243,00	32.477,40	5.070,00
900	30.285,00	39.226,30	5.070,00
1000	30.241,00	33.296,30	5.070,00

Tabla 7: Comparativa Func. Eval. – Avisos Fuera de Sección

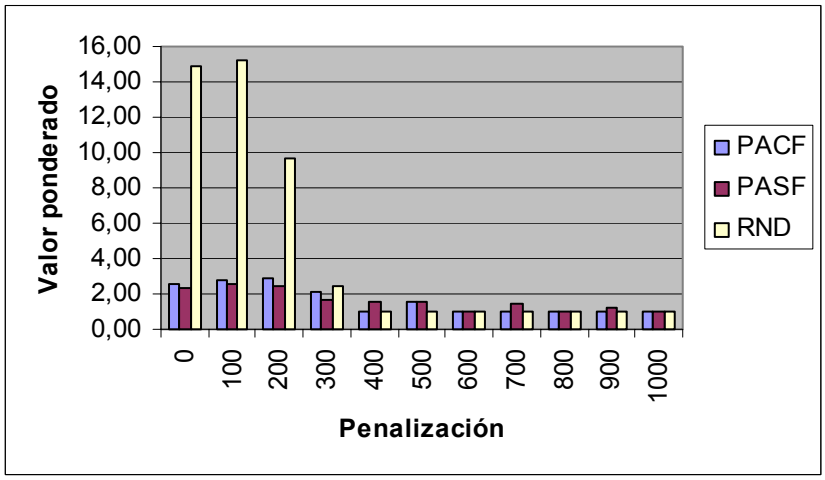

Figura 16: Comparativa Func. Eval. – Avisos Fuera de Sección

En este caso se puede apreciar que para valores muy bajos, se producen las anormalidades que estamos penalizando. Es decir, que con valores menores a 300, no se evita que algunos *avisos destacados* se posicionen en páginas previas al encabezado. Por tal motivo, un valor alto para esta penalización es lo más conveniente. En el cuadro se puede observar que el valor mínimo es alcanzado en todos los lotes para una penalización de 800.

Distancia:

Este parámetro ajusta el valor de la penalización de la distancia entre los *avisos destacados* y su *encabezado de sección* correspondiente. A mayor distancia, mayor penalización. A diferencia del caso de avisos fuera de sección, este valor afecta a los *avisos destacados* ubicados en páginas posteriores a su encabezado. Considerando que una buena solución tendría todos sus avisos en esta condición, la mayoría de los *destacados* serían alcanzados por esta penalización, salvo aquellos que se ubiquen en la misma página que el encabezado. En consecuencia, el valor elegido para la misma no puede ser muy alto.

Veamos los resultados arrojados por las pruebas:

Distancia	PACF	PASF	RND
100	1.165,40	2.153,70	1.187,00

200	1.965,30	2.928,70	1.687,00
300	2.743,40	12.610,70	2.187,00
400	3.627,40	22.258,80	2.687,00
500	21.692,70	40.189,00	3.187,00
600	57.712,50	104.878,10	3.687,00
700	60.219,80	101.945,80	4.176,10
800	103.576,00	109.044,20	12.278,10
900	107.448,40	117.285,30	4.953,20
1000	107.432,40	117.574,20	66.345,50

Tabla 8: Comparativa Func. Eval. – Distancia

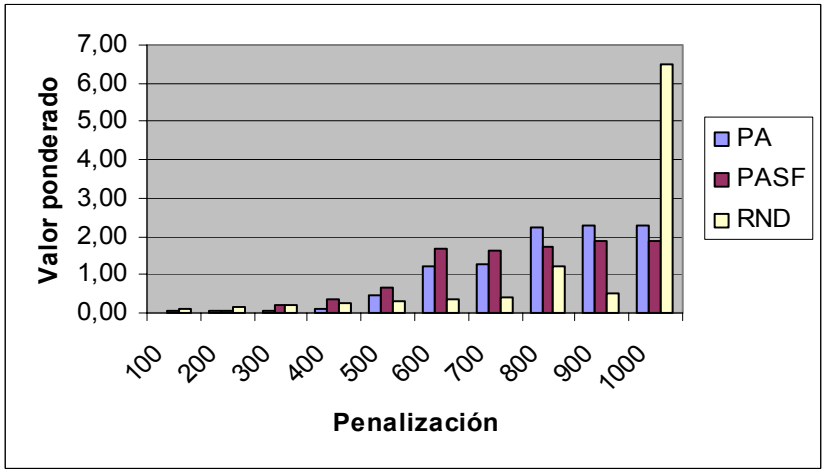

Figura 17: Comparativa Func. Eval. – Distancia

Como era de esperarse, valores altos de este parámetro causan que, con el afán de acercar lo más posible los avisos al comienzo de la sección, algunos *destacados* pasen a estar en páginas previas a su encabezado, quedando así fuera de sección. Para este ajuste, valores de penalización menores o iguales a 100, son los que han arrojado los mejores resultados.

6.2.2 Ajuste de parámetros de Búsqueda Tabú

Porcentaje de Vecinos:

El ajuste de esta variable fue realizado en dos etapas, una con un valor de retención tabú bajo, y otra con un valor más alto.

Los resultados obtenidos para una y otra combinación fueron distintos, por lo que podemos ver que existe una dependencia entre ambos parámetros.

%Vecinos	PACF	PASF	RND
20	30.499,00	101.573,10	5.373,60
40	42.252,60	95.062,70	5.321,00
60	84.288,30	89.322,60	11.987,80
80	85.258,50	89.318,00	45.321,80
100	91.356,00	89.318,00	45.321,80

Tabla 9: Comparativa BT – Porcentaje de Vecinos (retención baja)

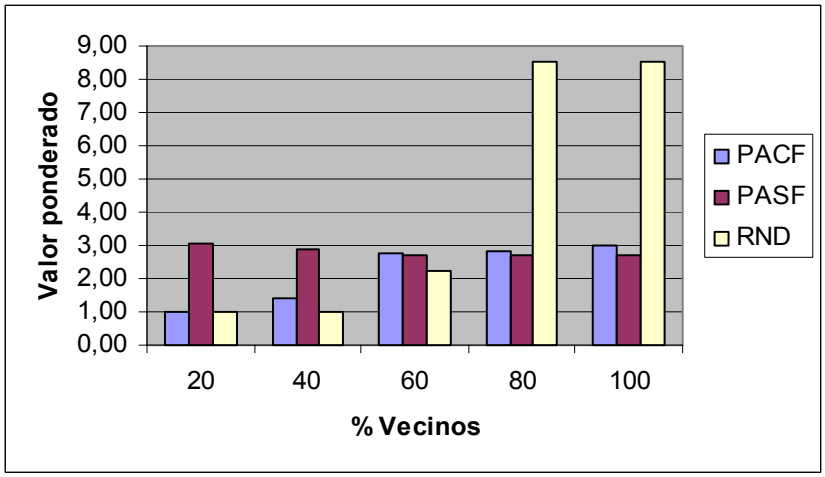

Figura 18: Comparativa BT – Porcentaje de Vecinos (retención baja)

En el caso de retenciones tabú bajas, el lote de pruebas **PASF** no genera resultados buenos, independientemente de la cantidad de vecinos analizada. Los demás lotes, tienen convergencia para valores bajos, pero en el caso de analizar un número mayor que el 40% de los vecinos, las soluciones generadas evidencian alguna anormalidad no permitida en una instancia final.

A simple vista, el análisis de mayor cantidad de vecinos debería conducir a mejores soluciones, incluso la convergencia debería producirse en menor cantidad de iteraciones, pero en este caso no se cumple.

La razón de este comportamiento, obedece a que la retención es demasiado baja, con lo que el algoritmo no es capaz de trasladarse de un valle a otro en casos en que la barrera a sortear sea lo suficientemente alta.

De todas maneras, en el caso específico de los lotes **PACF** y **RND**, con valores de 20% y 40% obtenemos muy buenos resultados en varias de las corridas, y esto se debe a la componente aleatoria de este tipo de algoritmos. La selección aleatoria de un grupo relativamente pequeño de vecinos, permite al algoritmo moverse hacia algún elemento lo suficientemente lejano del óptimo local, como para poder desde allí, saltar hacia otro espacio de búsqueda. Esta forma de diversificación no había sido considerada hasta el momento en nuestro trabajo. De todas maneras, y como es de esperar, la obtención de buenos resultados no se puede garantizar para todas las corridas de un mismo lote, siendo además, muy dependiente del problema.

Cuando consideramos retenciones tabú más altas, los resultados parecen ser más lógicos. Veamos los valores obtenidos para los lotes de prueba:

%Vecinos	PACF	PASF	RND
20	55.380,20	47.391,00	35.944,50
40	30.508,70	30.183,00	6.133,20
60	30.423,60	29.786,70	5.629,50
80	30.503,00	29.354,30	5.605,80
100	30.177,00	29.512,40	5.452,60

Tabla 10: Comparativa BT – Porcentaje de Vecinos (retención alta)

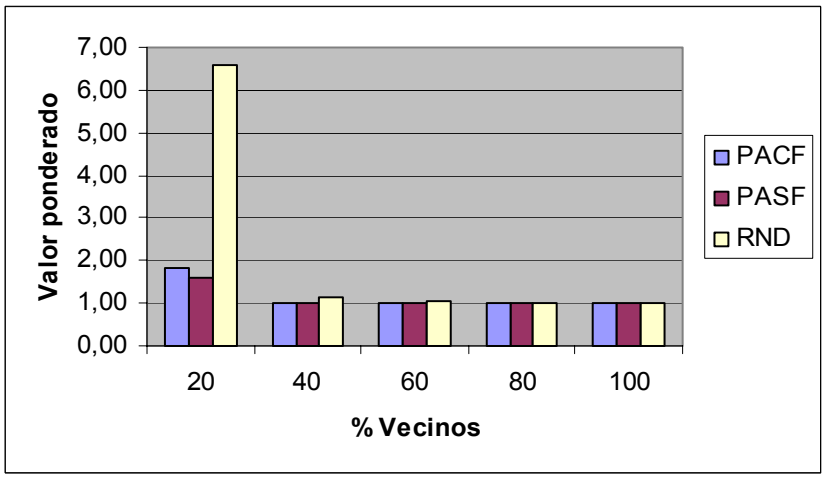

Figura 19: Comparativa BT – Porcentaje de Vecinos (retención alta)

Aquí vemos que la diversificación de la búsqueda está dada por la retención y no por el azar. De hecho, el peor caso para los tres lotes de prueba corresponde al porcentaje más bajo de análisis de vecinos. Podemos deducir entonces, que una lista tabú larga no es compatible con una selección azarosa de muy pocos vecinos, pero si da buenos resultados a partir de un valor que podría rondar el 30% o 40%.

Como conclusión, podemos decir que este valor está ligado a la retención tabú, y que para listas lo suficientemente largas, analizar el 40% de los vecinos puede ser una buena solución. A pesar de que los resultados obtenidos para valores superiores al 40% son también muy buenos, elegimos este porcentaje debido a que la cantidad de soluciones analizadas es considerablemente menor, reduciendo notoriamente el tiempo de corrida del algoritmo.

Retención de la lista tabú:

Como hemos explicado anteriormente, la retención era seleccionada en forma aleatoria dentro de un rango de valores. Este intervalo, definido por [a$\sqrt{d}$, b$\sqrt{d}$], es ajustado en base a los valores de "a" y "b", siendo "d" la cantidad de *avisos destacados* involucrados.

Estos valores (que por cuestiones prácticas están dados en términos de porcentaje), fueron variados en un rango entre 0 y 1000.

A/B	PACF	PASF	RND
0-100	15199,00	89948,60	2870,00
100-200	15199,00	74978,20	2870,00
200-300	15223,00	15734,00	2887,60
300-400	15207,00	15854,00	3010,00
400-500	15259,00	15733,00	3022,00
500-600	15215,00	15896,20	3255,20
600-700	15227,00	15957,60	3007,60
700-800	15369,00	16034,20	3192,40
800-900	15191,00	15903,60	3176,40
900-1000	15613,00	16415,80	3114,80

Tabla 11: Comparativa BT – Retención A y B

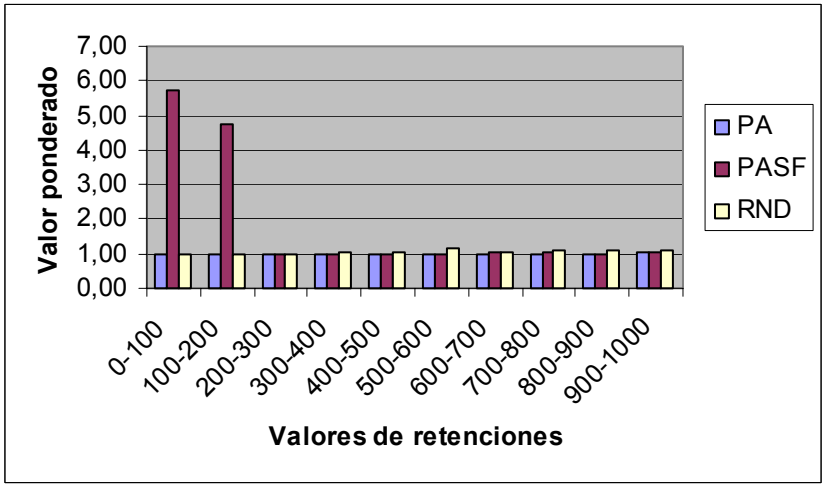

Figura 20: Comparativa BT – Retención A y B

En el gráfico podemos observar que el lote más afectado por este parámetro es el lote **PASF**, el cual, como hemos visto antes, para valores de retención bajos no llega a obtener una buena solución. Para un rango de valores de [200, 300], logramos muy buenos resultados en todos los casos, por lo que tomamos este rango como el mejor. Si traducimos estos valores a cantidad de iteraciones, tendríamos un rango aproximado de [15, 23] (considerando que para nuestros lotes, con ventanas de 20 páginas, se procesan aproximadamente 60 *avisos destacados*).

Cantidad de iteraciones:

La cantidad de iteraciones necesarias normalmente debe variar en función del tamaño del problema, de la *solución inicial*, y de la eficiencia del algoritmo.

Las pruebas se realizaron con los mejores parámetros de cada grupo, y una ventana de procesamiento de 20 páginas. Los siguientes gráficos nos dan una idea de la cantidad de iteraciones necesarias para cada lote.

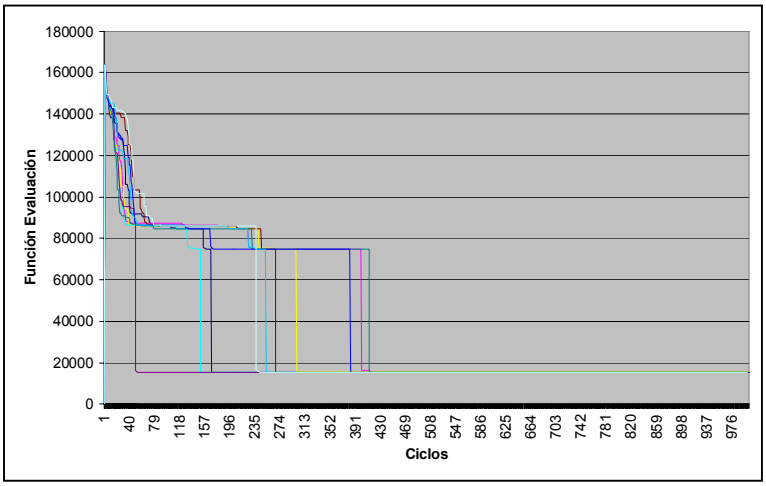

Figura 21: Comparativa BT - Convergencia Lote PACF

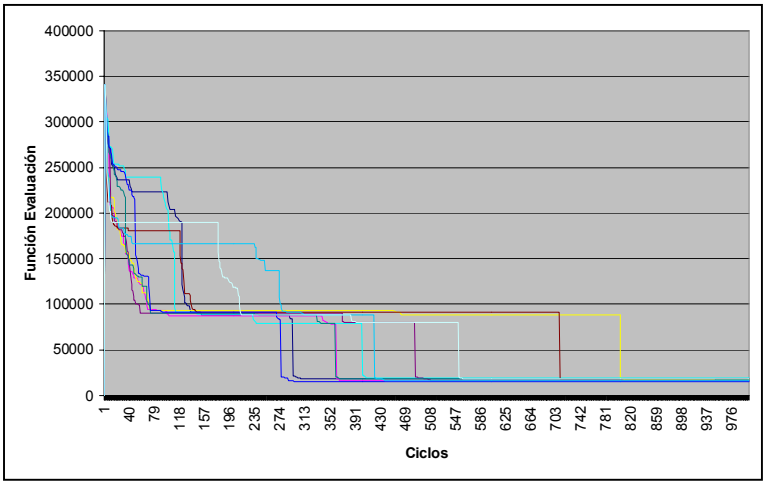

Figura 22: Comparativa BT - Convergencia Lote PASF

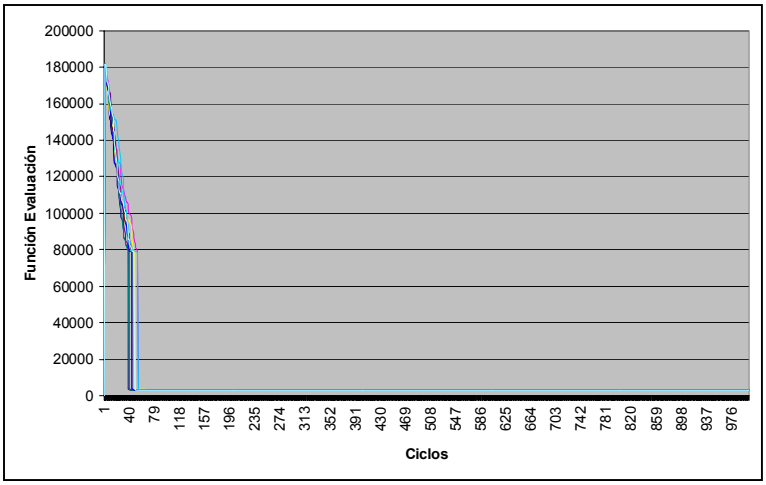

Figura 23: Comparativa BT - Convergencia Lote RND

Como se ve en los gráficos, la convergencia es bastante diferente para los tres lotes analizados, con lo que podemos ver la dependencia de los datos antes mencionada. La cantidad de iteraciones necesarias para cada lote son: 450, 800 y 50, y en todos los casos hemos tomado los peores valores. Otra variable que influye sobre la velocidad de convergencia es la cantidad de vecinos analizada. Como vimos anteriormente, para valores relativamente altos de retención tabú, un análisis de mayor cantidad de vecinos reduciría la cantidad de iteraciones necesarias, aunque los casos analizados en cada ciclo se verán incrementados. El porcentaje de vecinos analizados fue del 20% para las pruebas realizadas. Consecuentemente con lo explicado, si hubiéramos subido la cantidad de vecinos, las iteraciones hubieran sido menos, pero no así el tiempo de procesamiento.

Como conclusión, podemos decir que con 1000 iteraciones estamos "cubiertos" en la mayoría de los casos. Si redujéramos el tamaño del problema tomando ventanas de procesamiento de por ejemplo 10 páginas, sería factible utilizar un valor aún menor.

Tamaño de ventana:

El cambiar el tamaño de la ventana, produce principalmente la variación del tamaño de problema. Debido a que la lectura de los avisos a memoria se hace ventana por ventana, el tamaño de la publicación completa no es importante.

En el ajuste de parámetros realizado anteriormente, como se explicó al comienzo de este punto, se han utilizado ventanas de 20 páginas. De todas maneras, de nuestras pruebas preliminares, podemos decir que con tamaños de 10 páginas las soluciones obtenidas son igualmente buenas, y los tiempos de procesamiento son menores. Utilizar tamaños muy pequeños puede causar algún tipo de "anormalidad", debido principalmente a que, cuando se miden distancias entre *avisos destacados* y encabezados ubicados en distintas ventanas, la medición se hace en forma estimada. Esto es debido a la falta de información de la ventana que aun no está en memoria. Por esta razón, la recomendación es no utilizar valores muy inferiores a las 10 páginas.

6.2.3 Mejores parámetros obtenidos

En base a los ajustes realizados sobre los tres lotes de prueba, tratamos de obtener un conjunto de parámetros que puedan producir buenos resultados en general.

La siguiente tabla muestra los mejores valores obtenidos para los distintos lotes, y en la última columna, los valores que dedujimos como "recomendables":

	Lote PACF	Lote PASF	Lote RND	General
# Iteraciones	1.000	1.000	1.000	1.000
# Vecinos	20 %	20 %	40 %	40%
Retención (A/B)	200/300	200/300	200/300	200/300
Overflow	400	500	900	500
Burbujas	6	8	6	8
Fillers	1	7	2	2
Fuera de Sección	900	800	1000	800
Distancia	100	100	100	100

Tabla 12: Selección de los mejores parámetros

Estos parámetros son los que han sido utilizados de aquí en adelante, para las pruebas sobre las cuales se hace el análisis de los resultados.

6.3 Influencia de la Solución Inicial

Según la calidad del algoritmo implementado, la *solución inicial* puede tener influencia o no en la obtención de buenos resultados. La *solución inicial* utilizada para todas las pruebas anteriores se genera usando una heurística constructiva, tal como fue explicada en el punto 4.1.1. Para verificar el comportamiento del algoritmo con puntos de partida menos elaborados, hemos hecho modificaciones al programa para generar soluciones iniciales fijas. Estas nuevas soluciones iniciales generadas, son intencionalmente "malas" en cuanto a la disposición de avisos. Para ello, ubicamos todos los *destacados* en páginas posteriores a la última perteneciente a la ventana. Con esta distribución inicial, el algoritmo deberá ir moviendo los *destacados* poco a poco hacia las páginas iniciales, hasta llegar a una buena disposición de avisos.

Realizamos pruebas con la *solución inicial* alterada, para los tres lotes. Para poder comparar los resultados obtenidos, mostramos también la convergencia a partir de la *solución inicial* original. Los parámetros utilizados son los mismos para cada lote.

Los gráficos de convergencia obtenidos para las distintas corridas fueron los siguientes:

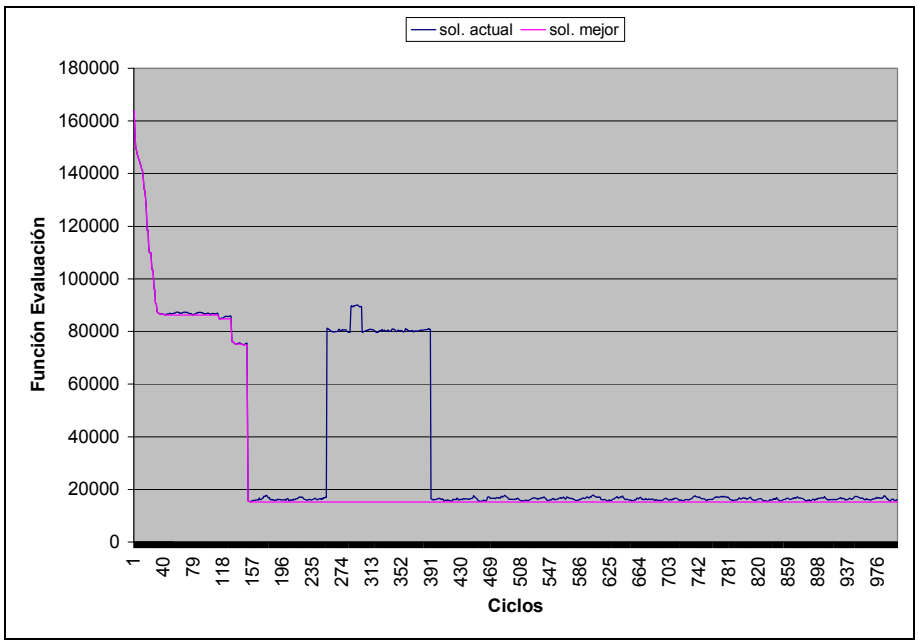

Figura 24: Sol. Inicial c/Heur. Constructiva – Lote PACF

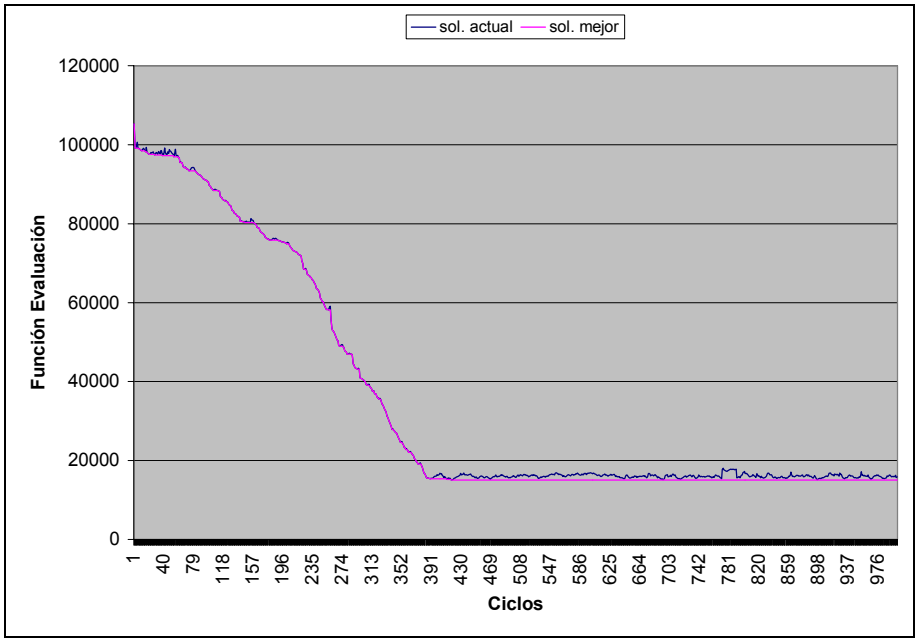

Figura 25: Solución Inicial "Mala" – Lote PACF

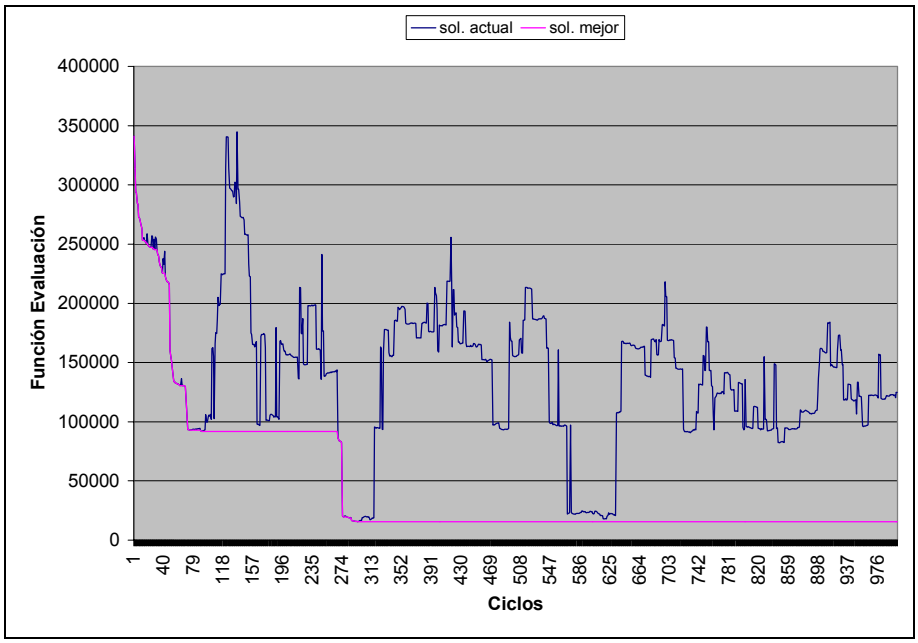

Figura 26: Sol. Inicial c/Heur. Constructiva – Lote PASF

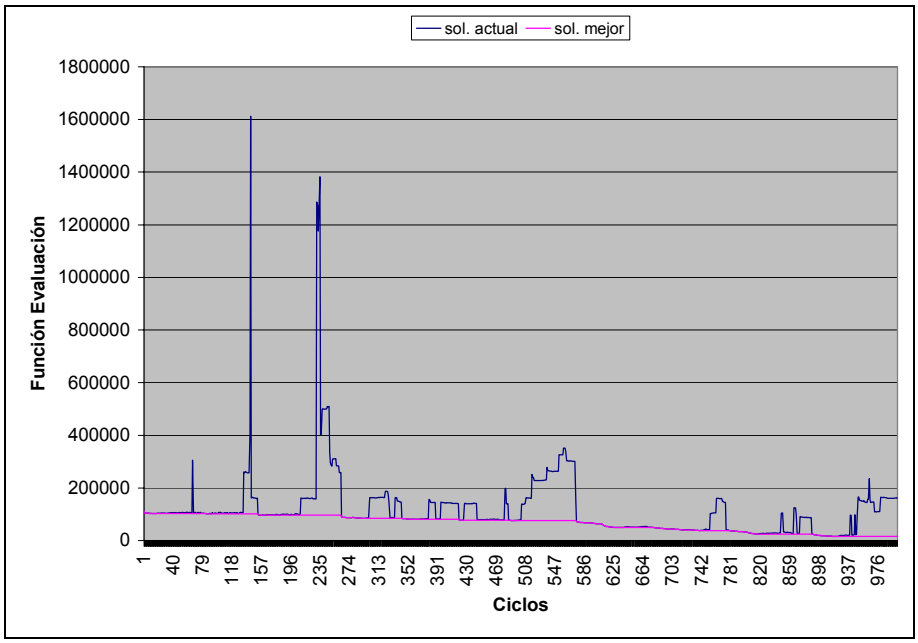

Figura 27: Solución Inicial "Mala" – Lote PASF

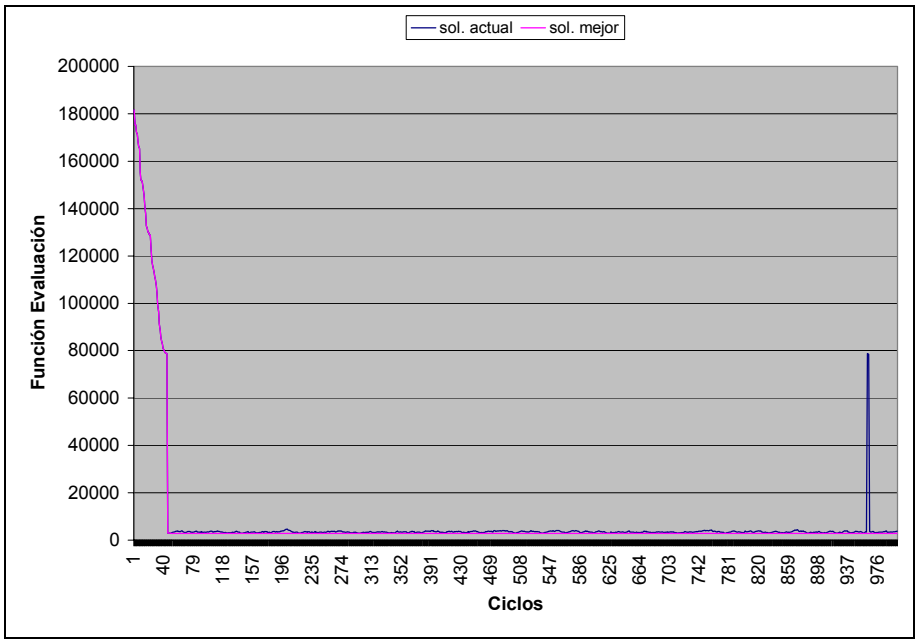

Figura 28: Sol. Inicial c/Heur. Constructiva – Lote RND

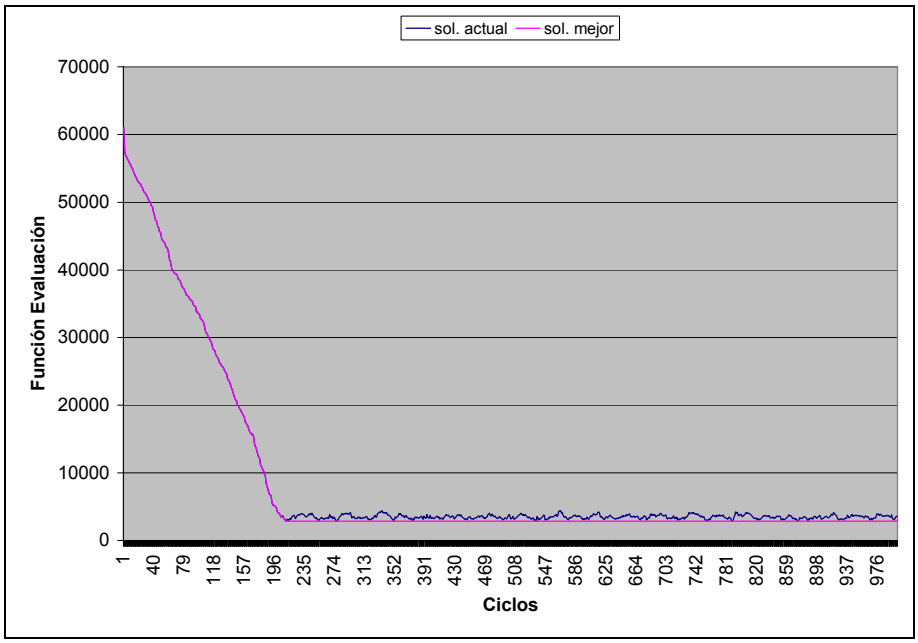

Figura 29: Solución Inicial "Mala" – Lote RND

Como puede apreciarse en los gráficos, y como era de esperar, con la *solución inicial* "mala" se requiere de muchas más iteraciones para alcanzar buenas soluciones. De todas maneras, se observa que la convergencia es siempre muy buena, y en todos los casos, se alcanzan resultados de la misma calidad que con la otra *solución inicial*.

Si analizamos más en detalle los gráficos, nos encontramos con cosas que a simple vista pueden parecer incoherentes: el valor de evaluación en el comienzo es menor para la *solución inicial* "mala". Esto se debe, a que por el simple hecho de dejar todos los *destacados* en páginas posteriores a la última de la ventana, no se producen overflows, *burbujas*, ni fuera de sección, mientras que con la *solución inicial*

original, este tipo de anormalidades es absolutamente factible. Esto reduce significativamente el valor de evaluación, ya que la única penalización aplicable es la correspondiente a la distancia y a los *fillers* generados al fluir los *lineales*. A partir de aquí, el algoritmo se focaliza en ir acercando los avisos a su sección, pudiendo observarse una convergencia más gradual que la observada en los otros casos.

Un hecho destacable, se puede apreciar en la **Error! Reference source not found.**, donde se ven picos extremadamente altos alcanzados por las soluciones intermedias, que pueden ser vistos como "barreras" a superar para poder pasar a una región del espacio con mejores soluciones.

6.4 Comportamiento de la Búsqueda Tabú

Como se ha dicho anteriormente, los resultados obtenidos han sido muy alentadores desde el punto de vista de la calidad de layout generado. Veamos un poco más en detalle el comportamiento en sí del algoritmo de *Búsqueda Tabú*. A lo largo de las pruebas realizadas hemos ido recolectando estadísticas, las cuales nos sirvieron para este análisis. Cabe aclarar que estas pruebas han sido corridas con el mismo conjunto de parámetros, los cuales habían sido seleccionados en el punto 6.2.3. Analicemos el siguiente cuadro:

	PACF		PASF		RND100		RND1000	
	Mejor sol.	Promedio	Mejor sol.	Promedio	Mejor sol.	Promedio	Mejor sol.	Promedio
Valor Solución	36.680	36.700	36.116	36.146	18.366	18.740	116.506	120.498
Total Ciclos	6.000	6.000	6.000	6.000	13.000	13.000	127.000	127.000
Ciclo mejor solución[1]	335	807	128	911	2.747	2.754	31.037	25.721
% Convergencia	5,58	13,44	2,13	15,19	21,13	21,19	24,44	20,25
Vecinos analizados	101.454	102.076	102.700	101.939	167.610	169.333	1.290.121	1.290.145
Posiciones Tabú	45.456	45.763	46.025	45.658	84.544	85.096	693.180	692.605
% Pos. Tabú	44,80	44,83	44,81	44,79	50,44	50,25	53,73	53,68
Crit. Asp. Aplicados	38	30	32	41	81	93	1.000	958
% Crit. Asp.	0,084	0,066	0,070	0,089	0,096	0,109	0,144	0,138
Mov. No mejora	3.075	3.073	3.042	3.063	6.358	6.421	61.024	61.023
% No mejora	51,25	51,22	50,70	51,05	48,91	49,39	48,05	48,05

Tabla 13: Estadísticas de Búsqueda Tabú

Aquí podemos observar varios de los aspectos del funcionamiento interno del algoritmo.

Con respecto a la convergencia, observamos que el ciclo en el que se obtiene la mejor solución es mucho menor al total de iteraciones del algoritmo. Tomando el peor de los promedios, se ha requerido el 21,19% de las iteraciones totales, lo que confirma nuestra teoría de que al reducir el tamaño de la ventana, la convergencia se produce más rápidamente. Para ventanas de tamaño 20, requeriría aproximadamente el 80%. Recordemos que la cantidad de vecinos procesados ha sido del 40%.

Si observamos el porcentaje de posiciones Tabú analizadas, vemos que ronda el 50%, con lo que podríamos deducir que, o la retención es demasiado alta, o bien, elegimos un criterio tabú demasiado amplio. Cuando estudiamos las distintas posibilidades de considerar tabú a los movimientos, vimos que el criterio elegido era el más restrictivo. Aquí podemos confirmar esta característica. De todas maneras, la retención utilizada fue de entre 15 y 23 ciclos, valores que a simple vista también parecen altos. Sin

[1] Acumulado para todas las ventanas procesadas.

embargo, gracias a esta gran cantidad de vecinos prohibidos, es que podemos diversificar la búsqueda y encontrar así los valles con los mejores resultados.

Ligado a la cantidad de vecinos tabú, podemos ver cuántas veces se ha cumplido el criterio de aspiración. Esta condición fue aplicable en muy pocas ocasiones (aproximadamente, a 1 de cada 1000 posiciones tabú evaluadas), lo que demuestra que el criterio utilizado para los movimientos prohibidos no estaba mal.

Por último, vemos que los movimientos de no mejora son en promedio de un 50%. Otra vez, aquí se puede presuponer que el porcentaje es demasiado alto. Pero teniendo en cuenta que la convergencia a la mejor solución encontrada se realiza en muy pocos ciclos, puede que no lo sea tanto.

6.5 Tiempos de procesamiento

Para tener una idea respecto de los tiempos insumidos, por el prototipo implementado, para la obtención de "buenas" soluciones, detallamos a continuación algunos valores de referencia.

El procesamiento de un lote de 50 páginas, con alrededor de 150 *destacados*, demora aproximadamente 2 minutos, con un tamaño de ventana de 10 páginas, 1000 iteraciones por ventana y 40% de vecinos a analizar. Con los mismos parámetros, la corrida de un lote de 1000 secciones (con alrededor de 1500 *destacados*), que generó aproximadamente 1140 páginas de información, demoró 25 minutos.

El hardware utilizado para estas pruebas de referencia ha sido una PC con procesador Pentium III – 700Mhz, con 128 MB de memoria RAM. La plataforma fue Windows 2000 edición profesional.

6.6 Calidad de los resultados obtenidos

6.6.1 Comparación con *Las Páginas Amarillas*

Como habíamos explicado anteriormente, uno de los lotes de prueba ha sido tomado de una guía telefónica real, con lo que podemos comparar los resultados obtenidos por nuestro algoritmo contra los publicados. El lote mencionado ha sido separado en dos lotes independientes, con la sola diferencia que en un caso, hemos fijado un *aviso destacado* en una página determinada obligando al algoritmo a romper el orden preestablecido.

Los casos analizados son referenciados a continuación como: **REAL**, para el caso de la extracción de la edición comercial de *Las Páginas Amarillas*; **PACF**, para nuestro lote semi-real con *aviso destacado* fijado; y **PASF**, para nuestro lote sin intervención manual.

Veamos primero un cuadro que resume las características obtenidas en los distintos casos:

	REAL	PACF	PASF
Páginas	51	51	51
Páginas en Overflow	0	0	0
Fuera de sección	0	0	0
% Fillers	1,60	1,72	1,69
% Burbujas	0,00	0,03	0,07
#Avisos a dist.0	28	25	24
# Avisos a dist.1	17	19	28
# Avisos a dist.2	27	27	18
# Avisos a dist.3	7	7	8
# Avisos a dist.4	12	17	17
# Avisos a dist.5	13	11	11
# Avisos a dist.6	3	11	11
# Avisos a dist.7	11	1	1
# Avisos a dist.8	2	2	2
# Avisos a dist.9	6	6	6

Tabla 14: Comparación de resultados con *Las Páginas Amarillas*

Aquí podemos observar claramente que la cantidad de páginas generadas es la misma, y que no se ubican avisos en *Overflow* ni fuera de sección para ninguno de los tres casos.

Con respecto al porcentaje de *fillers* generado, podemos decir que al parecer, en la publicación real hay un poco menos de espacio desperdiciado. Decimos "al parecer", ya que no nos fue posible medir en la publicación comercial, el desperdicio que es "disimulado" al hacer una *justificación vertical* de los *avisos lineales* en las columnas.

En el caso de las *burbujas*, estas no han sido observadas en *Las Páginas Amarillas*, pero sí se han producido en las publicaciones generadas por nuestro algoritmo. Entre las heurísticas se han mencionado algunas ideas que mejoran la disposición de avisos dentro de una página, reduciendo la cantidad de desperdicio de este tipo. Estas técnicas más avanzadas no han sido implementadas en nuestro prototipo. De todas maneras, la cantidad de *burbujas* es muy baja para los resultados obtenidos con nuestras pruebas (aproximadamente, 1 o 2 *burbujas* de 6 módulos de alto).

Para el caso de las distancias de los *avisos destacados* a su encabezado, observemos primero el siguiente gráfico:

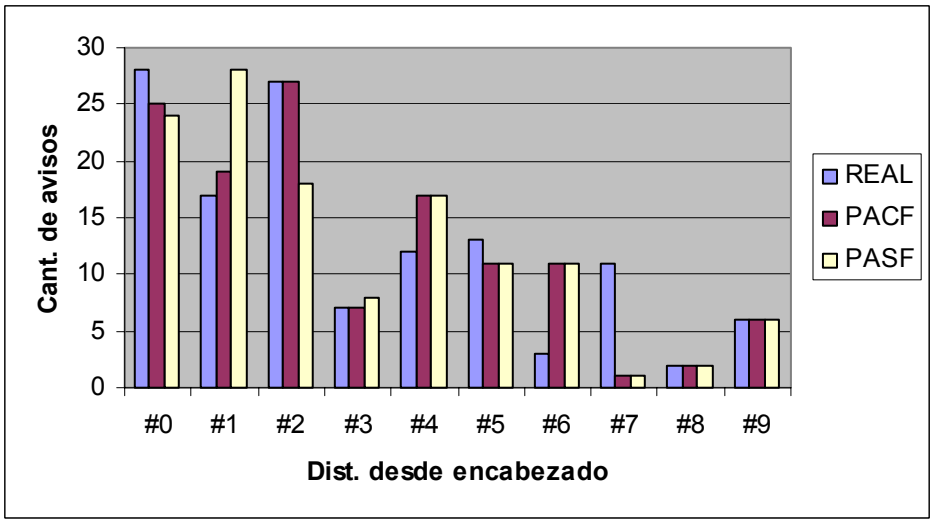

Figura 30: Distribución de avisos destacados en páginas

Aquí podemos observar que la distribución por distancias es bastante pareja. Podemos ver que en *Las Páginas Amarillas* existen más *avisos destacados* a distancia 0 (en la página del encabezado) que en nuestras pruebas, pero la diferencia es muy poca. Además, esta diferencia se compensa a 1 página de distancia, y en el caso de PASF, es superada.

Hemos detectado, analizando la publicación de *Las Páginas Amarillas*, algunos avisos fuera de orden. En algunos casos, un cambio en el orden de los *avisos destacados* puede provocar una mejor distribución de los mismos, en términos de distancia o de desperdicio generado. Estas tareas normalmente las realiza un operador en el proceso de control de calidad de la publicación. Obviamente, los resultados generados por nuestro algoritmo no han sido modificados manualmente.

En conclusión, podemos decir que nuestro algoritmo produce soluciones casi tan buenas como las observadas en la guía telefónica real, con la diferencia que para nuestras pruebas, no hubo intervención manual, mientras que en la publicación comercial, es evidente que así fue.

6.6.2 Análisis de los valores recolectados

En la Tabla 15 resumimos los resultados obtenidos para todos los lotes de prueba analizados. Los mismos corresponden a 10 corridas realizadas para las publicaciones completas. En este sentido, utilizamos para todas ellas los parámetros "standard" obtenidos previamente.

Como podemos observar claramente, en ninguno de los casos (tanto para las Mejores Soluciones obtenidas, como para el Caso Promedio) se produce *overflow* ni existen *avisos destacados* fuera de rubro.

Respecto del desperdicio generado (*fillers* y *burbujas*), en el caso de la publicaciones de *Las Páginas Amarillas*, el mismo no supera el 1.75 % en promedio. Para el caso de los lotes generados aleatoriamente (**RND100** y **RND1000**), debemos aclarar que el porcentaje de *Fillers* (aproximandamente del 3 %), puede deberse sobre todo a la gran proporción de *avisos lineales* "grandes" (de 10 filas). Sin embargo, en todos los casos, el porcentaje de *burbujas* generadas (correspondiente a "malformaciones" en el layout de las páginas), nunca supera el 0.25 %.

	PACF		PASF		RND100		RND1000	
	Mejor sol.	**Prom.**	**Mejor sol.**	**Prom.**	**Mejor sol.**	**Prom.**	**Mejor sol.**	**Prom.**
Valor Solución	36.680	36.700,00	36.116	36.146,00	18.366	18.740,20	116.506	120.497,80
#Páginas	51	51,00	51	51,00	115	115,00	1.139	1.139,80
Tam. Ventana	10	10,00	10	10,00	10	10,00	10	10,00
#Ventanas	6	6,00	6	6,00	13	13,00	127	127,00
Min Ciclos[2]	335	806,50	128	911,30	2.747	2.754,40	31.037	25.721,10
Min Ciclos/Vent[3]	55,83	134,42	21,33	151,88	211,31	211,88	244,39	202,53
#Págs Oveflow	0	0,00	0	0,00	0	0,00	0	0,00
% Fillers	1,72	1,72	1,69	1,69	3,12	3,10	2,96	3,02
% Burbujas	0,03	0,03	0,07	0,07	0,25	0,20	0,02	0,03
#Avisos F/Secc	0	0,00	0	0,00	0	0,00	0	0,00
#0	25	24,80	24	23,80	85	84,60	815	784,30
#1	19	19,20	28	28,20	93	88,70	612	643,70
#2	27	27,00	18	18,00	21	24,90	118	116,60
#3	7	7,00	8	8,00	3	3,80	9	8,50
#4	17	17,00	17	16,90	0	0,00	0	0,90
#5	11	11,00	11	11,10	0	0,00	0	0,00
#6	11	11,00	11	11,00	0	0,00	0	0,00
#7	1	1,00	1	1,00	0	0,00	0	0,00
#8	2	2,00	2	2,00	0	0,00	0	0,00
#9	6	6,00	6	6,00	0	0,00	0	0,00

Tabla 15: Resumen de los Valores Obtenidos

Otra característica que debemos recalcar es la velocidad con la que se produce la convergencia a la solución obtenida. Como podemos corroborar en la tabla anterior, el número de iteraciones promedio (para cada ventana) tanto de la Mejor Solución obtenida, como de la Solución Promedio, nunca supera las 300 iteraciones.

En el Apéndice B incluimos la salida completa de la mejor corrida obtenida para el lote **PASF**, que corresponde a la publicación extractada de *Las Páginas Amarillas* sin la fijación manual de *avisos destacados*.

Por último, en la Figura 31 y Figura 32, se observa la forma en que se agrupan los *avisos destacados* respecto del inicio de sección (para el caso de los lotes **RND100** y **RND1000**). Para el caso de **PACF** y **PASF**, esta distribución ya ha sido graficada en su comparación con la publicación real (Figura 30).

[2] Corresponde al número de iteración de la mejor solución obtenida (acumulado para todas las ventanas procesadas).

[3] Corresponde al número de iteración promedio de la mejor solución obtenida (para cada ventana).

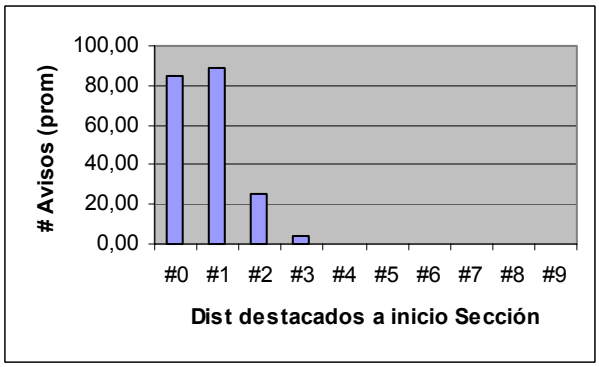

Figura 31: Distribución Destacados para lote RND100

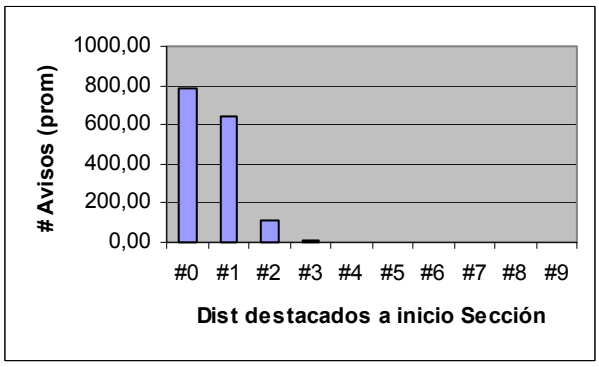

Figura 32: Distribución Destacados para lote RND1000

Con esta información, podemos concluir que los parámetros ajustados anteriormente generan "buenas" distribuciones, agrupando dentro de cada sección los *avisos destacados* hacia el inicio de la misma. Este es, precisamente, uno de los criterios o requerimientos que definen al problema tratado.

6.6.3 Análisis de los criterios gráficos

A continuación analizamos algunos de los resultados obtenidos, pero esta vez desde el punto de vista de los criterios gráficos considerados para el layout de las páginas.

En la Figura 33 podemos observar un ejemplo de los dos casos en que se torna necesario rellenar con *Fillers*. Al final de las columnas 1, 3 y 5 de la página 15 (derecha), vemos cuándo el algoritmo introduce *Fillers* debido a que "el siguiente *aviso lineal* a ubicar no cabe" en el espacio disponible. Por otro lado, en el caso de las columnas 2 y 4 de la misma página, se insertan *Fillers* debido a la restricción que impide colocar un *encabezado de sección*, si no cabe al menos un *aviso lineal* debajo suyo. Cabe aclarar que la necesidad de generar esta clase de rellenos podría evitarse, en algunos casos, realizando justificación vertical al fluir los *avisos lineales*.

Como contrapartida, en la otra página (14), podemos ver cómo "se las ingenia" el algoritmo implementado para distribuir los *avisos destacados* y los *avisos lineales* sin producir ninguna clase de desperdicio.

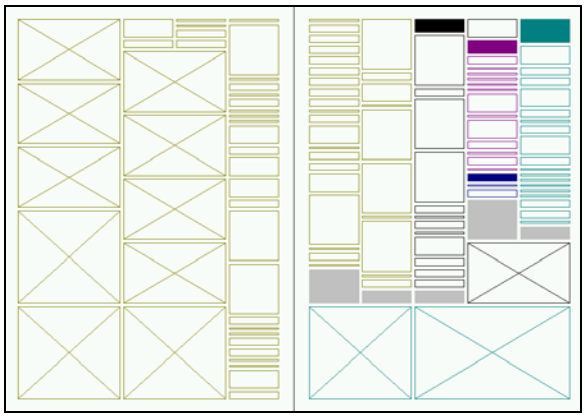

Figura 33: Págs. 14/15 del lote PASF – Generación de Fillers

Al mismo tiempo, si miramos ambas páginas como "un todo", este es un ejemplo concreto de un buen resultado obtenido mediante la heurística utilizada para ubicar los *avisos destacados* en forma de "U".

Respecto de la generación de *burbujas*, en la Figura 34 ilustramos el caso de la página 49 (de la salida del lote **PASF** detallada en el Apéndice B). A la izquierda de la figura podemos apreciar cómo posibles malformaciones en la ubicación de los *avisos destacados* en la página da lugar a "huecos" de desperdicio. Esto podría mejorarse si reubicáramos los *avisos destacados* como se muestra a la derecha. Cabe aclarar que en este caso dicha mejora sería sólo estética ya que, como podrá verse en la página 50 de dicha salida, el siguiente *aviso lineal* a ubicar (de 10 filas) no entraría en esta página. Por lo tanto, el espacio remanente sería rellenado con *fillers*.

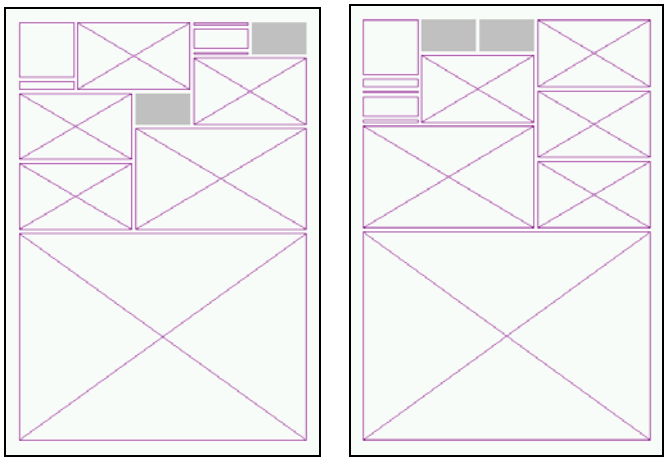

Figura 34: Página 49 del lote PASF, con y sin burbuja

Finalmente, en la Tabla 16 podemos ver los resultados obtenidos con dos de las variantes heurísticas implementadas para la distribución de *avisos destacados* en la página.

De los valores recolectados en las pruebas realizadas podemos concluir que los mejores resultados, al menos en términos del desperdicio generado (*burbujas*), corresponden a las distribuciones en las que los avisos son "acostados", esto es, agrupados horizontalmente.

	Acostando		Apilando	
	Mejor sol.	**Prom.**	**Mejor sol.**	**Prom.**
Valor Solución	36.116	36.146,00	36.284	36.346,00
Min. Ciclos[4]	128	911,30	116	568,50
#Páginas	51	51,00	51	51,00
#Págs. Overflow	0	0,00	0	0,00
% Burbujas	0,07	0,07	0,49	0,47
#Avisos F/Sección	0	0,00	0	0,00
#0	24	23,80	30	29,70
#1	28	28,20	22	22,10
#2	18	18,00	16	16,20
#3	8	8,00	10	10,00
#4	17	16,90	16	15,70
#5	11	11,10	12	12,30
#6	11	11,00	11	11,00
#7	1	1,00	1	1,00
#8	2	2,00	2	2,00
#9	6	6,00	6	6,00

Tabla 16: Comparación "Acostando" vs. "Apilando"

[4] Corresponde al número de iteración de la mejor solución obtenida (acumulado para todas las ventanas procesadas).

7 CONCLUSIONES

7.1 Efectividad de la estrategia elegida

En función de los resultados obtenidos, podemos decir que la estrategia implementada puede ser aplicada al **Problema de la Paginación de Avisos Clasificados**.

En este sentido se han obtenido muy buenos resultados, tal como se los ha analizado en el Capítulo 6, para distintos lotes de prueba. Las soluciones obtenidas, en el caso del lote extraído de *Las Páginas Amarillas*, son de muy buena calidad, e incluso similares a las impresas en la guía. Más aún, si consideramos que en la publicación comercial se evidencia la intervención manual del operador.

Uno de los mayores problemas con el que nos encontramos, fue la dificultad de conseguir datos reales con los cuales hacer comparaciones. De hecho, la única forma fue extraer manualmente una muestra de una publicación comercial.

Con respecto específicamente a nuestro algoritmo, podemos concluir que la representación de los datos elegida, fue fundamental para una implementación eficiente. Sobre esta representación, hemos verificado que es posible la obtención de "buenas" soluciones mediante la aplicación de un algoritmo de *Búsqueda Tabú*. Por otro lado, podemos decir que los componentes elegidos para la *metaheurística* implementada (función movimiento, criterios de movimientos tabú, criterio de aspiración, etc.), han demostrado tener un buen desempeño en conjunto. Incluso la técnica propuesta resultó ser lo suficientemente robusta como para no depender de la calidad de la *solución inicial* implementada.

Si bien está visto que la cantidad de soluciones analizadas por el algoritmo es elevada, los tiempos de corrida del mismo, son totalmente aceptables para la implementación de un sistema interactivo, en el cual haya un operador esperando por los resultados frente a la pantalla.

La convergencia a buenas soluciones, tal como se ha visto en el capítulo anterior, se produce generalmente en forma muy rápida. Si aprovechamos esta característica para la selección de un criterio de parada más elaborado, los tiempos de corrida podrían reducirse aún más. En este sentido, la posibilidad de establecer un mínimo para el valor de la solución, tal como fue implementado en nuestro prototipo, es una forma simple de acotar los tiempos de respuesta, sin detrimento de la calidad de la solución obtenida.

En nuestro trabajo, hemos obtenido un conjunto de parámetros de referencia. Los mismos han demostrado ser lo suficientemente amplios para generar muy buenos resultados en todos los lotes testeados, considerando incluso que éstos tienen características diferentes. Si bien no podemos probar su efectividad en el 100% de los casos, podemos decir que son una buena medida a tomar como referencia, y realizar algún ajuste en caso que sea necesario.

Como conclusión final, podemos decir que la técnica de *Búsqueda Tabú*, tal como fue implementada, es perfectamente aplicable al **Problema de la Paginación de Avisos Clasificados**. La buena calidad de las soluciones generadas por nuestro algoritmo (sin necesidad de intervención manual), facilitaría considerablemente las tareas del operador. Asimismo, su intervención, introduciendo algunos cambios puntuales, siempre podrá dar un valor agregado al producto final.

7.2 Posibles mejoras y trabajos futuros

Respecto de la ubicación o distribución de los *avisos destacados* en la página, podrían realizarse las siguientes mejoras o extensiones:

- Combinar las heurísticas implementadas: combinaciones de los métodos "apilar" y "acostar" podría llevarnos a obtener mejores resultados.
- Desarrollar nuevas heurísticas: sería factible implementar alguna clase de método exhaustivo, o bien, algún tipo de búsqueda local a nivel de página.
- Implementar mecanismos de "reconocimiento de bloques": de esta forma se pueden tratar dos o más *avisos destacados* como un único objeto y reubicarlos como tal.

Respecto de la *Búsqueda Tabú*, podrían probarse y analizarse algunas extensiones al método general como ser:

- Incluir atributos de frecuencia: para poder incluir características de uso de memoria de largo plazo.
- Implementar Soluciones Elite: para el mantenimiento de buenas soluciones obtenidas a lo largo de la búsqueda, para posteriores profundizaciones.
- Ampliar el Criterio de Parada: tener en cuenta la cantidad de iteraciones de no mejora como alternativa para la finalización.

Existen además ciertas funcionalidades que, por escapar del objetivo de este trabajo, no fueron consideradas en la implementación del prototipo. Entre ellas, podríamos mencionar:

- Generar automáticamente elementos ancla: los lectores podrían identificar más rápidamente (por orden alfabético), una entrada que se encuentre como *aviso destacado* dentro de la publicación.
- Justificar verticalmente los *avisos lineales*: esto permitiría "disimular", el espacio de desperdicio en las columnas.
- Manejar contenido dentro de los avisos: incluir gráfica y texto dentro de los elementos (*avisos destacados*, *lineales* y de relleno), mejoraría la orientación del usuario del sistema.
- Generar salida para impresión en formato standard: archivos con formato Postscript y/o EPS.
- Incluir la posibilidad de iniciar o retomar el proceso de paginación desde una página cualquiera (y no necesariamente desde la primera página).

Bibliografía

[A89] H. Ahonen: **"Knowledge-based integration of Newspaper Design and Layout"**, Lasers in Graphics ´89 (1989).

[A94] H. Ahonen: **"Embedded optimization tasks in a pagination problem"**, SOBRAPO, XXVI Simposio Brasileiro de Pesquisa Operacional, Florianópolis (1994).

[AV89] H. Ahonen, P. Viertio: **"Knowledge-based Pagination"**, In Proceedings of the SCAI '89 of The Second Scandinavian Conference on Artificial Intelligence, Tampere - Finland (1989).

[CHLKOT] Chew, Hong-Gian, Lian, Koh, Ong, Tan: **"ALEXIS: An Intelligent Layout Tool for Publishing"**, In Proceedings of the Sixth Annual Conference on Innovative Applications of Artificial Intelligence, Seattle, pag. 41-47 (1994).

[CM94] R. E. Campello, N. Maculan: **"Algoritmos e heuristicas - Desenvolvimento e avaliaçao de performance"**. Editora de Universidade Federal Fluminense (1994).

[G86] F. Glover: **"Future Paths for Integer Programming and Links to Artificial Intelligence"**, Computers and Operations Research 5 (1986).

[G95] F. Glover: **"Tabu Search Fundamentals and Uses"**, Working Paper, University of Colorado, Beulder (1995).

[GL93] F. Glover y M. Laguna: **"Tabu Search"**, Modern Heuristics Techniques for Combinatorial Problems, Blackwell Scientific Publications (1993).

[GNG96] Graf, Neurohr, Goebel: **"YPPS - A Constraint-Based Tool for the Pagination of Yellow Page Directories"**, Technical Report, German Research Center for Artificial Intelligence (1996).

[GTD93] F. Glover, E. Taillard y D. De Werra: **"A User's Guide to Tabu Search",** Annals of Operations Research, Vol. 41, Pag. 3-28 (1993).

[HMWB] Haranda, Mikako, Witkin, Baraff: **"Interactive Physically-Based Manipulation of Discrete/Continuous Models"**, In Proceedings of SIGGRAPH'95, Los Angeles. Pag. 199-208 (1995).

[HTD] A. Harch, E. Taillard, D. De Werra: **"Tabu Search"**. Basic ideas of tabu search.

[IIDYFT] Iwai, Isamu, Doi, Yamaguchi, Fukui, Takebayashi: **"A Document Layout System using Automatic Document Architecture Extraction"**, In Proceedings of CHI'89, Austin (1989).

[JMPS97] R. Johari, J. Marks, A. Partovi, S. Shieber: **"Automatic Yellow-Pages Pagination and Layout"**, Journal of Heuristics (1997).

[K95] K. Lagus: **"Automated pagination of the generalizad newspaper using simulated annealing"**, Helsinki University of Technology (1995).

[KGV83] S. Kirkpatrick, C.D. Gelantt y M.P. Vecchi: **"Optimization by Simulated Annealing"**, Science 220 (1983).

[KP81] Knuth, Donald E. And Michael F. Plass: **"Breaking Paragraphs into Lines"**. Software-Practices and Experience 11. Pag. 1119-1184 (1981).

[L94] M. Laguna: **"A Guide to implementing Tabu Search"**, Investigacion Operativa 4 (1994).

[L95] M. Laguna: **"Tabu Search Tutorial"**, II Escuela Latinoamericana de Verano de Investigacion Operativa, Rio de Janeiro (1995).

[P81] M. F. Plass: **"Optimal Pagination Techniques for Automatic Typesetting Systems"**, PhD Thesis, Satanford University (1981).

[PA99] **"Las Páginas Amarillas" de Capital Federal (Edición 1999-2000)**, Grupo Telecom, páginas 945 a 995.

[R93] C. R. Reeves: **"Modern Heuristic Techniques for Combinatorial Problems"**. Blackwell Scientific Publications, London (1993).

[RJHMES] Rosenking, Jeffrey, Howard, Marmostein, Eva, R. Soccio: **"A generic System for Directory Pagination"**, In Proceedings of the IEEE/ACM International Conference on Developing and Managing Expert System Programs (1991).

[VALNY] T. Veijonen, H. Ahonen, V. Leikola, J. Niku-Paavola, J. Yla-Jaaski: **"Automatic Page Layout for Electronic News Distribution"**, In Proceedings of the IASTED/ISMM Conference on Distributed Multimedia Systems and Applications, Honolulu - Hawaii (1994).

[WW94] Weitzman, Louis and Kent Wittenburg: **"Automatic Presentation o Multimedia Documents Using Relational Grammars"**, In Proceedings of the Second Annual ACM Conference on Multimedia, San Francisco. Pag. 443-451 (1994).

[YA90] J. Yla-Jaaski, H. Ahonen: **"Knowledge-based newspaper pagination"**, Proceedings of the SPIE on Applications of Artificial Intelligence VIII, Orlando - Florida (1990).

Apéndice A: Contenido del material digital

A continuación enumeramos el contenido de material digital entregado en CD:

- Instalador del prototipo PAGINATION: "\Pagination"
- Documento de Tesis (este mismo documento): "\Documentacion\Tesis.pdf"
- Presentación de Tesis: "\Documentacion\Presentacion.ppt"
- Lotes de Prueba: "\Lotes de Prueba"
- Resultados: "\Resultados"
- Acrobat Reader 5: "\Acrobat5"

Apéndice B: Lote de prueba de *Las Páginas Amarillas* (PASF)

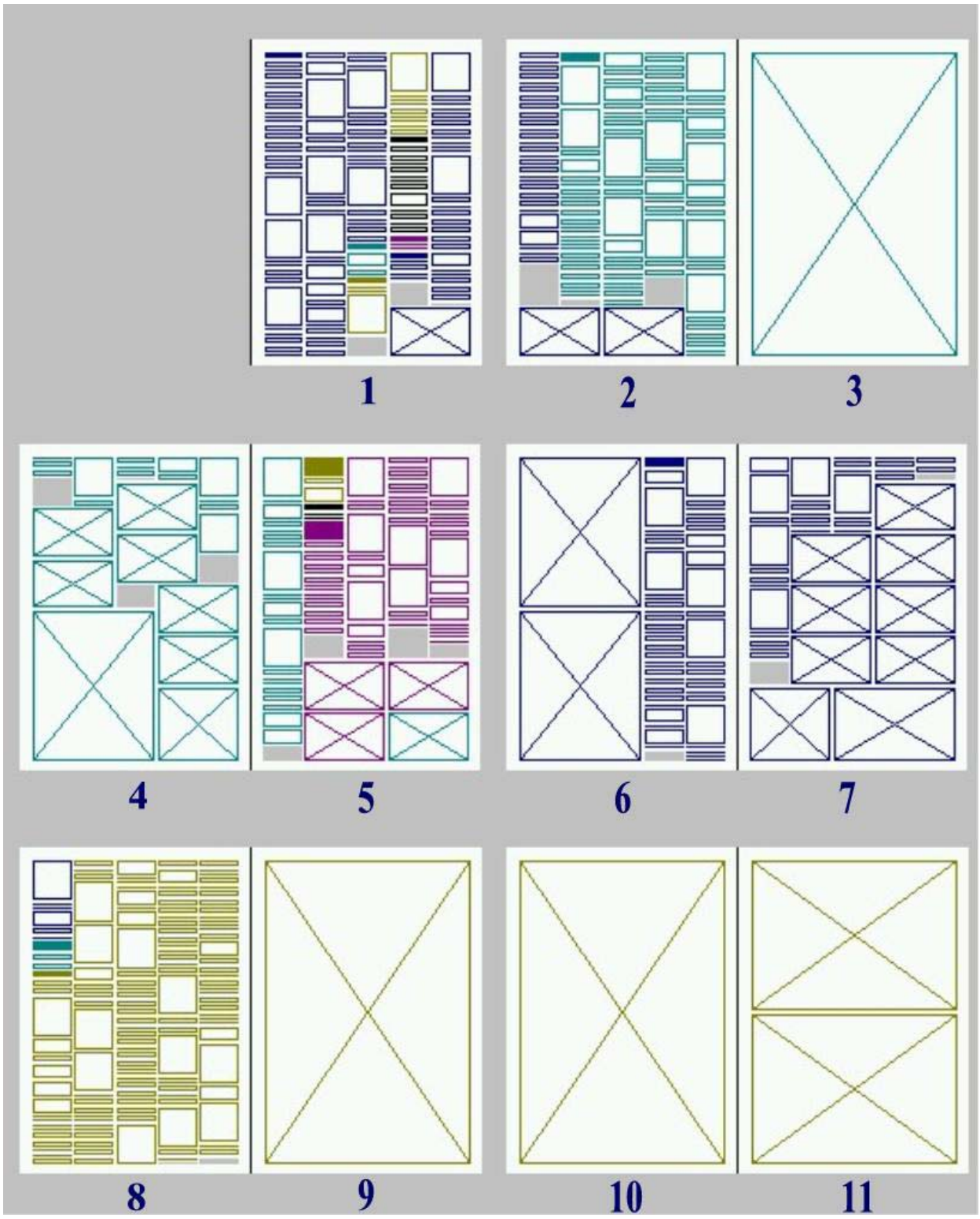

Figura 35: Lote PASF (Pags. 1-11)

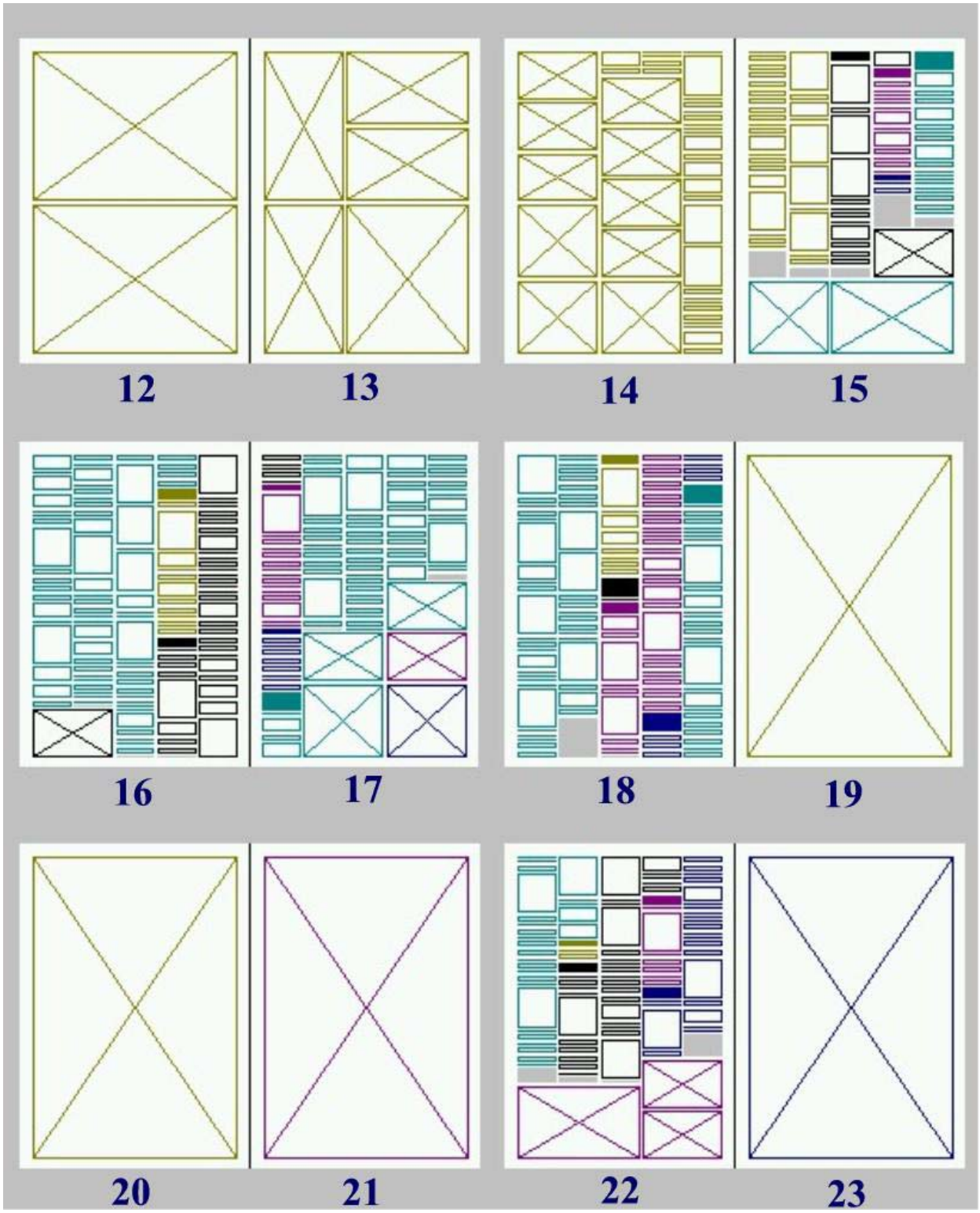

Figura 36: Lote PASF (Pags. 12-23)

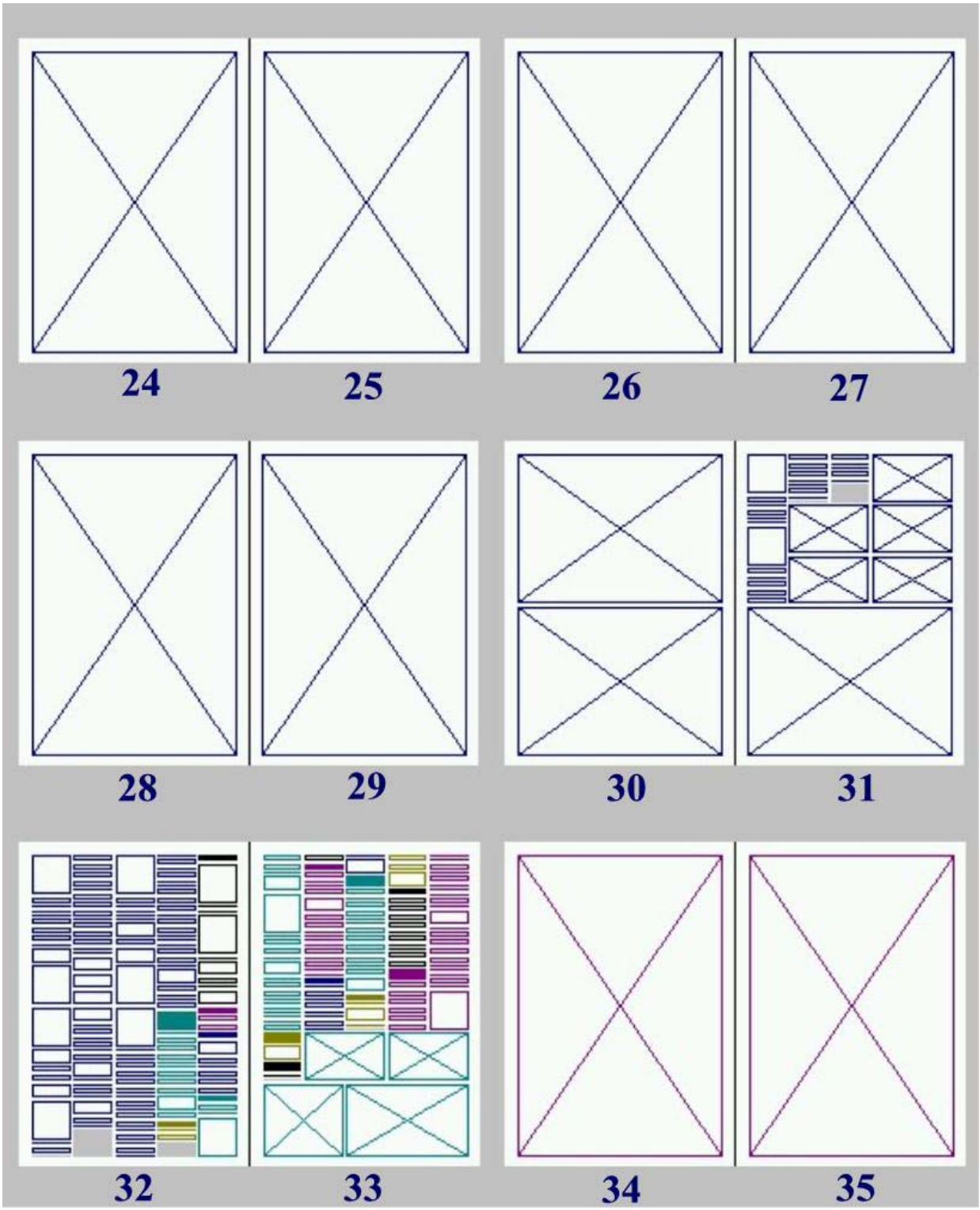

Figura 37: Lote PASF (Pags. 24-35)

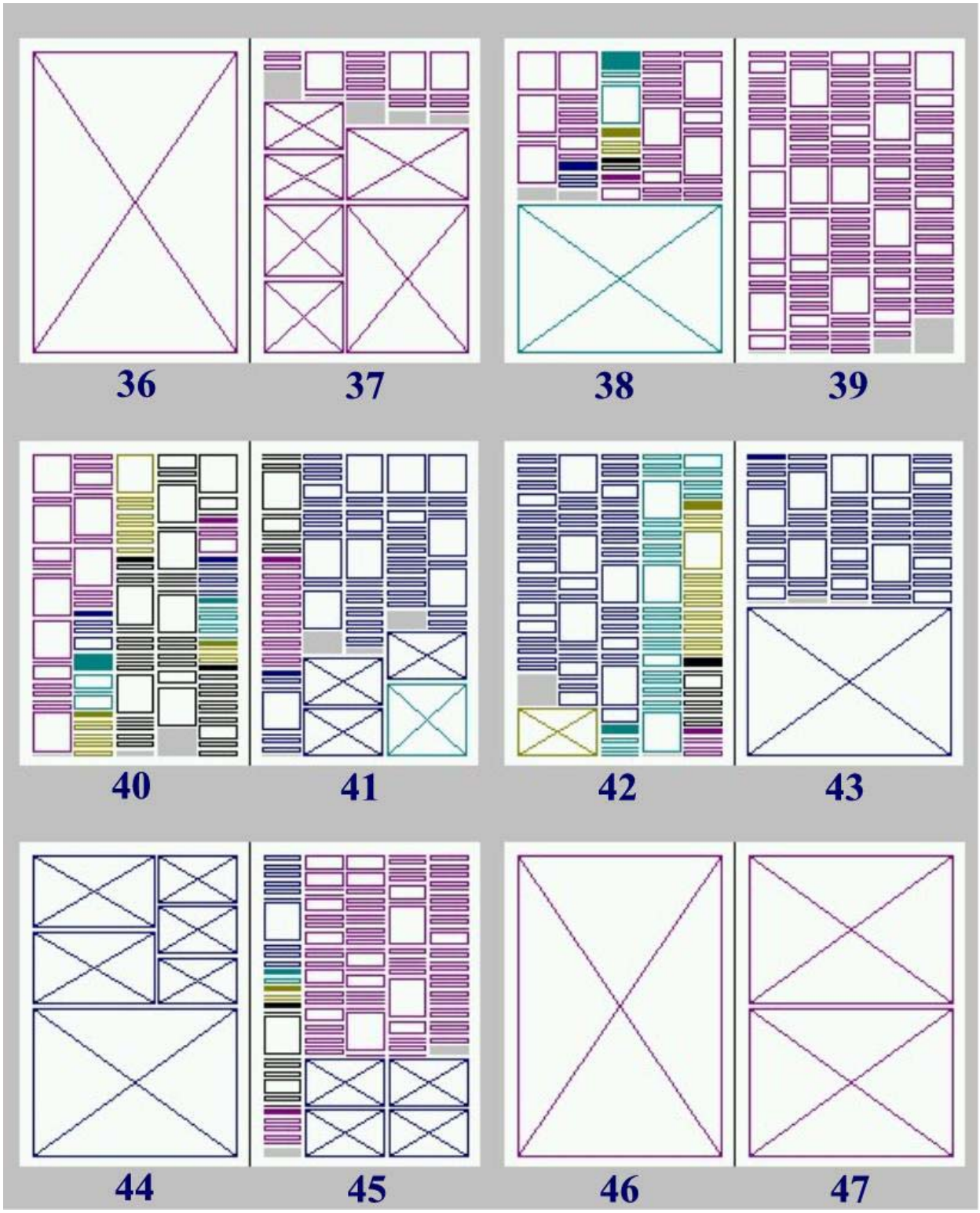

Figura 38: Lote PASF (Pags. 36-47)

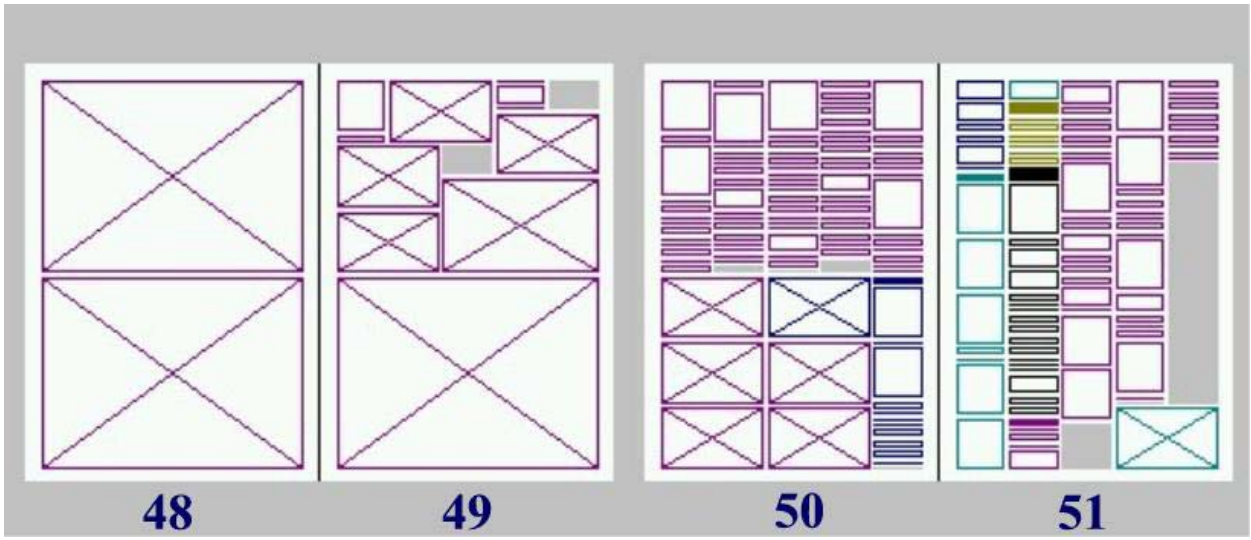

Figura 39: Lote PASF (Pags. 48-51)

Apéndice C: Funcionalidad del prototipo

A continuación se describen las principales funciones del prototipo desarrollado y la forma de operarlo:

Crear una publicación

Mediante este comando el usuario genera la estructura necesaria dentro del sistema para almacenar una nueva publicación. Para la misma, requiere de todos los archivos (con extensión .IN) con los datos de todos los elementos de su nueva publicación. El primer paso en la ejecución de este comando es la selección del archivo SECTION.IN (archivo que contiene todos los rubros o secciones de la publicación). Luego, el usuario, debe escribir el nombre que le quiere asignar a la nueva publicación junto al resto de los parámetros de la misma. Una vez realizado esto, el sistema genera la siguiente estructura:

Una carpeta con el nombre de la publicación ubicada por debajo de la carpeta de la aplicación
Copia todos los archivos .IN desde donde fue seleccionado el archivo SECTION.IN por el usuario hacia la nueva carpeta creada en el punto anterior.
Crea un archivo (pagination.INI) en donde se almacenan los parámetros de esta publicación.

Abrir una publicación

Genera el ambiente para levantar desde archivo la publicación seleccionada por el usuario.

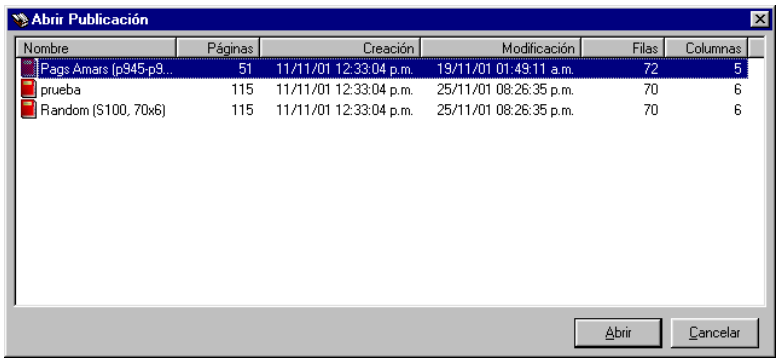

Figura 40: Interfase Gráfica - Abrir Publicación

Cerrar una publicación

Desaloja de memoria la estructura necesaria para contener una publicación

Paginar

Llama al Motor para ejecutar las funciones de paginación y diagramación de la publicación actual de memoria. Es posible especificar una página tope para este proceso o que contemple todas las páginas de la publicación. También, se puede elegir ver o no la ventana de procesamiento de la paginación.

Buscar una página

Esta función permite encontrar una página de una forma sencilla. El usuario ingresa el número de página a buscar y la aplicación posiciona la primer página de la grilla de páginas con la planchuela que contiene

a la misma, en caso de que las páginas subsiguientes superen la disponibilidad de la pantalla. Caso contrario, la página buscada estará dentro de alguna de las planchuelas desplegadas en la pantalla.

Recorrer secuencialmente la publicación

Mediante la barra de scroll (a la derecha de la pantalla), el usuario puede moverse por la publicación, de a n páginas por vez, donde n es:

- Cantidad de página de una fila: con scroll corto
- Cantidad de páginas en la pantalla: con scroll largo

Modificar parámetros de la publicación

Se despliega un cuadro de diálogo, en donde el usuario puede modificar los siguientes datos correspondientes a la publicación:

- Número de filas
- Número de columnas
- Gutter[5] horizontal
- Gutter vertical

Modificar parámetros de la página

Se despliega un cuadro de diálogo, en donde el usuario puede modificar los siguientes datos correspondientes a las páginas de la publicación:

- Tipo de página: A3, A4, letter, oficio, etc. o personalizada
- Alto y ancho de la página en el caso de ser de tipo de personalizada
- Orientación de la página: horizontal o vertical
- Márgenes: superior, inferior, externo, interno

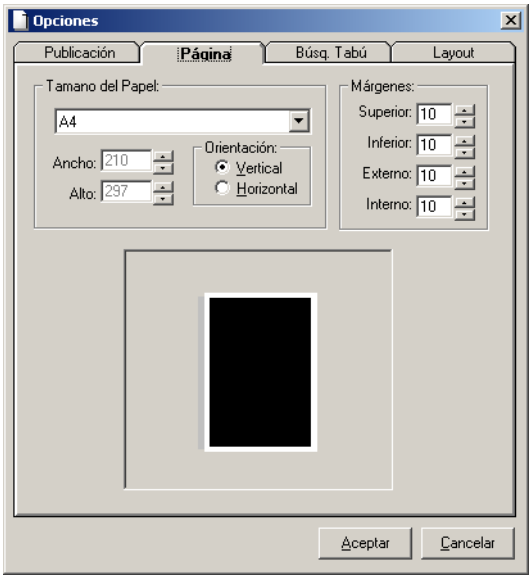

Figura 41: Interfase Gráfica - Opciones de la Página

[5] Gutter: Espacio entre avisos

Modificar parámetros de *Búsqueda Tabú*

Se despliega un cuadro de diálogo, en donde el usuario puede modificar los parámetros correspondientes al mecanismo de *Búsqueda Tabú*:

- Tamaño de ventana
- Cantidad máximo de ciclos
- Cantidad máximo de vecinos
- Retención en lista tabú (A y B)
- Valor mínimo de la solución

Y los correspondientes a las penalidades para la función evaluación:

- *Overflow* (según área y constante)
- Violación preferencia par / impar (según área y constante)
- *Bubbles* (según área)
- *Fillers* (según área)
- Fuera de sección: avisos delante de *headers* (según número de avisos y constante)
- Fuera de sección: avisos en secciones posteriores (según número de avisos y constante)
- Distancia a inicio de sección (según número de avisos y constante)

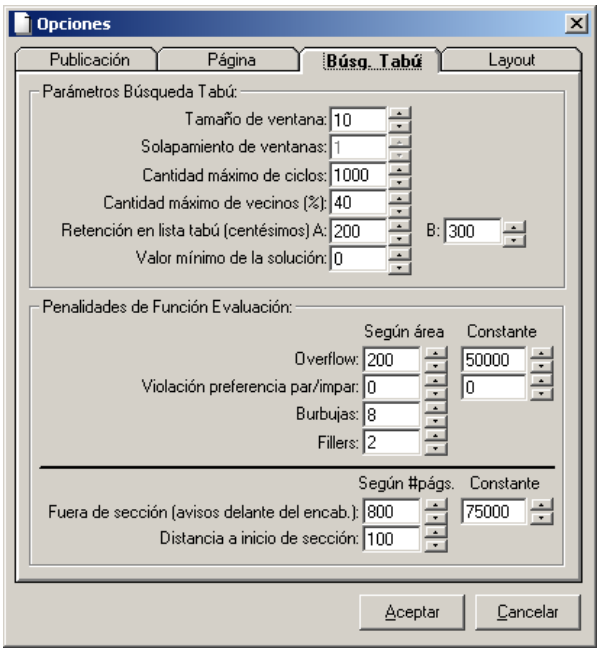

Figura 42: Interfase Gráfica - Opciones de Búsqueda Tabú

Modificar parámetros de Layout

Se despliega un cuadro de diálogo, en donde el usuario puede modificar los parámetros correspondientes al estilo de layout de páginas:

- Alineación de *avisos destacados* para páginas pares e impares: desde el exterior o interior
- Profundidad del apilamiento: máxima (apilar) o mínima (acostar)
- Cantidad mínima de *fillers* debajo del *encabezado de sección*

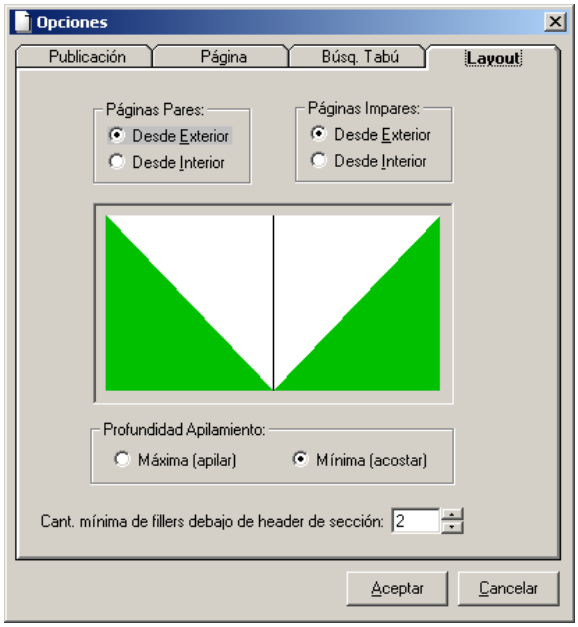

Figura 43: Interfase Gráfica - Opciones del Layout

Desplegar un mosaico de páginas

Mediante esta herramienta, el usuario puede generar una grilla de n cuadrantes de ancho por m cuadrantes de alto en los que cada cuadrante representa a una planchuela (dos páginas). Por lo tanto, es una forma sencilla de partir la pantalla para ubicar una grilla de pares de páginas.

Cambiar propiedades de avisos *destacados* y de avisos de relleno

Sobre un *aviso destacado* o de relleno, mediante el menú contextual (botón derecho del mouse) y seleccionando la opción de propiedades, o bien realizando doble click, se despliega el siguiente cuadro de diálogo:

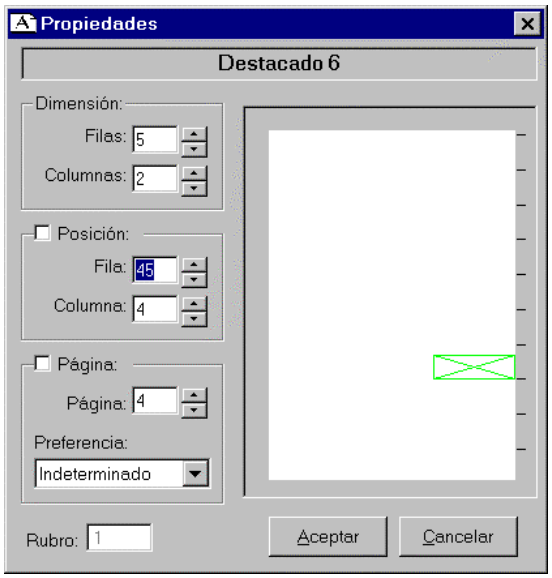

Figura 44: Interfase Gráfica - Propiedades del Aviso

En esta ventana se puede modificar las propiedades:

- Posición: fila y columna
- Número de página y preferencia (indeterminado, páginas pares o páginas impares)

O bien se podría fijar/desfijar la posición y/o fijar/desfijar la página. Estos cambios también pueden ser realizados directamente desde el menú contextual.

También se muestra sin poder alterarse, las dimensiones del elemento (filas y columnas) y el rubro al que pertenece. Y en la ventana de la derecha, se despliega una página reducida de muestra con un avance en la visualización del elemento según sus propiedades.

Modificación de posición de un aviso destacado o aviso de relleno

La posición de un *aviso destacado* o aviso de relleno puede ser modificado de las siguientes tres formas:

- Mediante drag & drop[6]: para mover la posición del elemento solamente dentro de la misma planchuela. Para realizar esta operación se requiere que la página sea de un tamaño suficientemente grande y que el usuario presione la tecla CTRL. al comenzar el proceso de drag.
- Mediante cut & paste[7]: para mover el elemento a otra página conservando la posición original. Se opera desde el menú contextual de los elementos de pantalla.
- Mediante modificación de propiedades: posición dentro de la página y número de página

Creación de aviso de relleno (filler)

Es posible la creación de un aviso de relleno o *filler*. Para esto, se requiere seleccionar el correspondiente comando desde el botón dentro de la barra de herramientas o vía menú y luego "dibujar" con el mouse el cuadrante que lo representa.

Identificación de pertenencia a una sección

Cuando se realiza un click de mouse sobre un *aviso destacado*, *aviso lineal* o *encabezado de sección*, se pueden visualizar todos los elementos que están en pantalla y que pertenecen a la misma sección del elemento "marcado". Esta función es útil para determinar el alcance de las secciones.

Monitoreo de posición

Con el movimiento del mouse sobre la zona de las páginas de la publicación es posible ir monitoreando las coordenadas (fila y columna) dentro de una página, el número de página y elementos con sus principales propiedades. Además se muestra el rango de las páginas presentados en pantalla y la cantidad de páginas totales de la publicación.

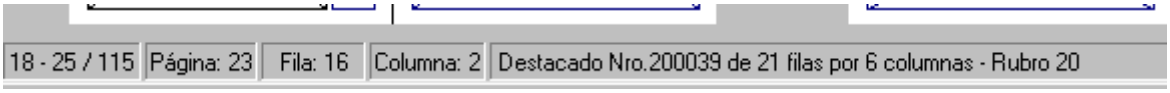

Figura 45: Interfase Gráfica - Línea de estado

[6] Drag & drop: mecanismo de "agarrar y soltar" objetos de la pantalla.

[7] Cut & paste: mecanismo de "cortar y pegar" objetos de la pantalla.

Selección de elemento

Ante posibles solapamiento de elementos dentro de una página, es posible alternar secuencialmente la selección de los distintos elementos solapados mediante la secuencia de teclas “CTRL + Click”.

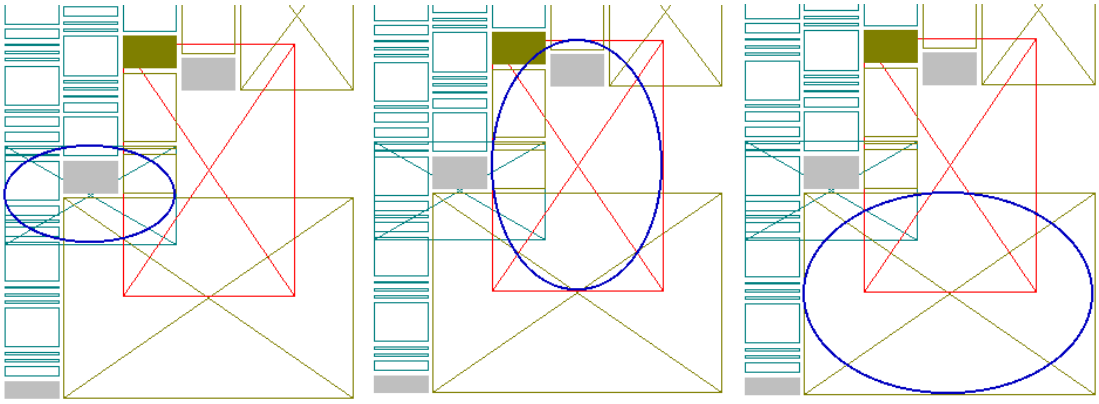

Figura 46: Interfase Gráfica - Selección de Avisos superpuestos

Overflow

Existe la posibilidad que luego de un proceso de Paginación, queden elementos (*avisos destacados* o de relleno) en *overflow*. Esto significa que estos elementos no pudieron ser ubicados dentro de la página por falta de espacio. Para ser identificada, la página que los contiene, se remarca con un cuadrante de color rojo en su borde. Con el comando *Overflow*, accedido desde el menú contextual, es posible desplegar una ventana que visualiza a todos los elementos de la página en este estado. Seleccionando un elemento de la lista y realizando “doble click” es posible editar sus propiedades, y así cambiarlas para poder ubicarlo en alguna página.

Apéndice D: Relevamiento de productos comerciales

SCS/ClassPag de SCS (Software Consulting Services)

http://www.newspapersystems.com/classpag/

SCS/ClassPag es una herramienta de paginación diseñada para componer un gran número de páginas de avisos clasificados en un corto período de tiempo. SCS/ClassPag cumple con este propósito utilizando criterios de colocación predefinidos por el usuario.

Los usuarios transfieren los avisos clasificados desde el sistema de ingreso de órdenes a SCS/ClassPag. El programa lee la información, acepta la reserva de espacios para noticias u otros elementos, y luego página la sección de acuerdo con la configuración suministrada.

Luego de un proceso inicial, es posible agregar o remover avisos propios de la publicación, rellenos o avisos desplegados y posteriormente adecuar, revisar o volver a paginar la sección en muy poco tiempo.

Una vez terminada una página, el usuario puede imprimirlas en PostScript, u opcionalmente, exportarlas como archivos EPS hacia aplicaciones de diagramación de páginas como GN3 o QuarkXPress.

Características:

- Produce páginas completas con avisos clasificados, desplegados, logotipos, gráficos y encabezados; tabulaciones impuestas e imágenes de páginas múltiples.
- Crea y coloca páginas automáticamente.
- Incluye un procesador de composición, con la funcionalidad de componer automáticamente desde la base de datos elementos dependientes del contenido, como por ejemplo índices.
- Diagrama secciones más compactas, ahorrando espacio.
- Página la sección de Clasificados de atrás hacia delante y viceversa.
- Permite cambios en el interlineado de *avisos lineales* y *destacados* de manera que se distribuyan uniformemente en las secciones.
- Localiza avisos rápidamente, con sólo ingresar algunos caracteres.
- Imprime páginas completas con *avisos destacados*, logotipos, gráficos, y titulares.
- Ajusta automáticamente documentos EPS de tamaño inapropiado.
- Permite ver los avisos, encabezados, titulares, y páginas en modo WYSIWYG.
- Prueba *avisos destacados* de acuerdo al tamaño, fuentes, colores, etc.
- Utiliza colores para identificar tipos de avisos y conflictos potenciales.

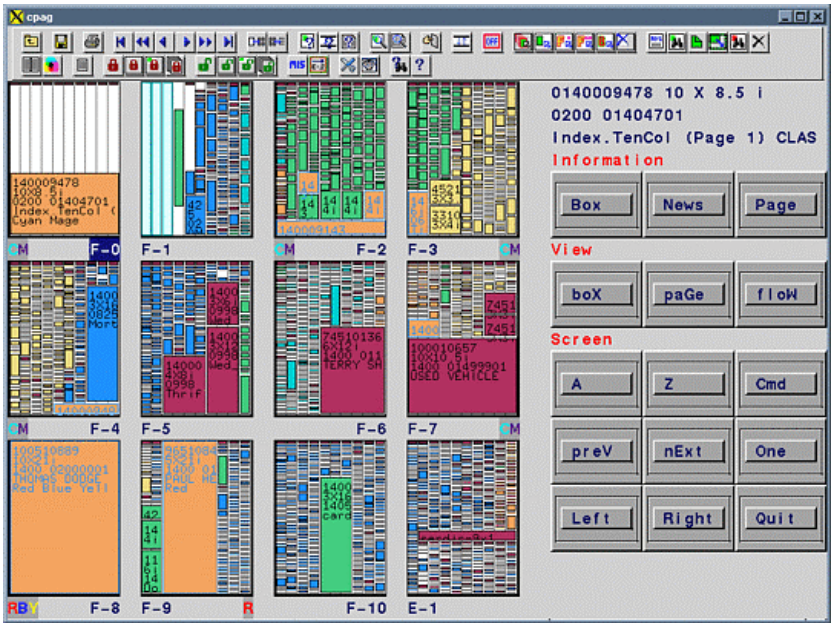

Figura 47: Interfase de usuario de CSC/ClassPag

VIP Intelligent Pagination Software de VTT Information Technology

http://www.vtt.fi/tte/samba/projects/vip/

VIP automatiza el proceso de colocar el material de publicidad en las paginaciones de periódicos, revistas y directorios. Esta herramienta considera las restricciones en la estructura y la disposición de las páginas, y optimiza el uso del espacio inteligentemente.

Es usado para el armado de las páginas amarillas del directorio telefónico de Gran Bretaña y Finlandia, además del armado de numerosos diarios.

Se pueden manejar múltiples ediciones de una publicación. Hay alternativas predefinidas disponibles para la dirección y modo de la paginación, alineación y estructuras de páginas. El sistema presenta la estructura total de la publicación así como también las páginas individuales y las extensiones. El usuario puede controlar el proceso con herramientas visuales interactivas.

Para la lógica de layout, aplica métodos avanzados de búsqueda derivados de tecnologías de inteligencia artificial combinados con conocimiento del proceso mismo.

Características:

- Corre sobre plataforma Unix
- Interfase de usuario bajo ambiente gráfico (X Window System)
- Flexible configuración de estilos de página y estructuras de diarios
- Paginación automática hacia delante y hacia atrás
- Diferentes estilos de layout de páginas
- Modo de llenado completo y parcial de avisos en las páginas
- Soporta una gran variedad de posición de avisos
- Herramienta interactiva flexible
- Visualización WYSIWYG de archivos PostScript con avisos
- Reservación de áreas especiales
- Configuración de librería de *fillers*
- Interfases para visualizar y clasificar avisos
- Simulación de página de salida

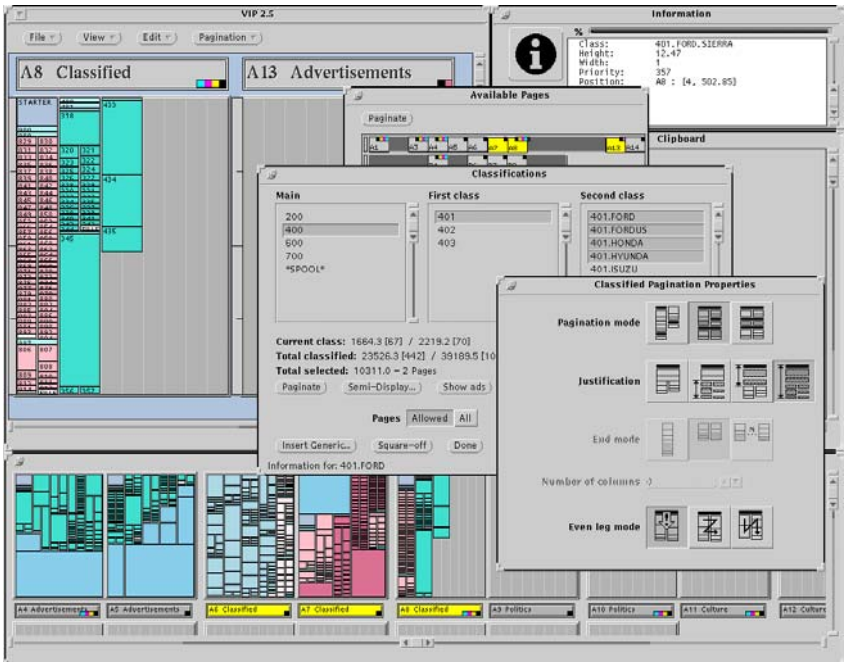

Figura 48: Interfase de usuario de VIP

Calligramme Directory

http://www.calligramme.com/homeus.htm

Calligrame Directory ha sido desarrollado en Francia en colaboración con editores de Páginas Amarillas para la paginación de directorios telefónicos.

Lee archivos de texto conteniendo las entradas (avisos y otros elementos) y compone todos estos elementos automáticamente produciendo archivos PostScripts de las páginas.

Algunos de los principales usuarios son: Pagine Italia (Milan), Belgacom (Bruselas), CMS (Paris), ESCM (Toulouse), ESSOR (Paris), ODA (Francia), Publicom (Buenos Aires), SNAT (Paris), Emap (Paris), Telelistas (Rio de Janeiro).

Características:

- Colocación de *avisos destacados* según la especificación de la publicación (posición en página) y según la sección a la que pertenece.
- *Avisos lineales* colocados según el flujo de entradas.
- Interfase gráfica: posiciona todos los elementos acordes a las reglas predefinidas, permitiéndole al usuario a que realice modificaciones.
- Elementos cortados en dos columnas.
- Cajas de referencias: genera automáticamente cajas de referencia para *avisos destacados* conteniendo información como tamaño y posición del aviso sobre la página y otras características.
- Generación automática de *avisos de relleno*.

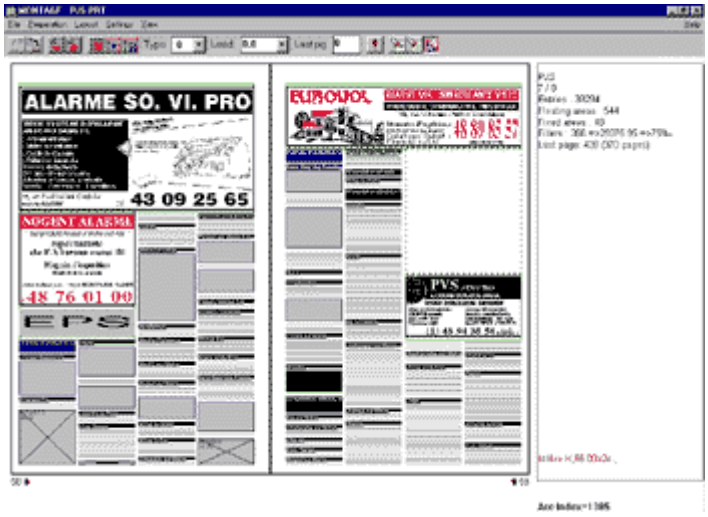

Figura 49: Interfase de usuario de Calligramme Directory

Page Planer & Pagination de NP Solutions

http://www.npsolutions.co.nz/Pagination.html

Forma parte de una suite de productos que apunta a la solución integral en el manejo de información de publicaciones de diarios y revistas.

Características:

- Corre sobre plataforma Windows.
- Criterios de preferencia para estilos de layout de páginas.
- Múltiples secciones y publicaciones.
- Distribución manual o automática de avisos clasificados.
- Exportación de la publicación hacia aplicaciones para su retoque gráfico.

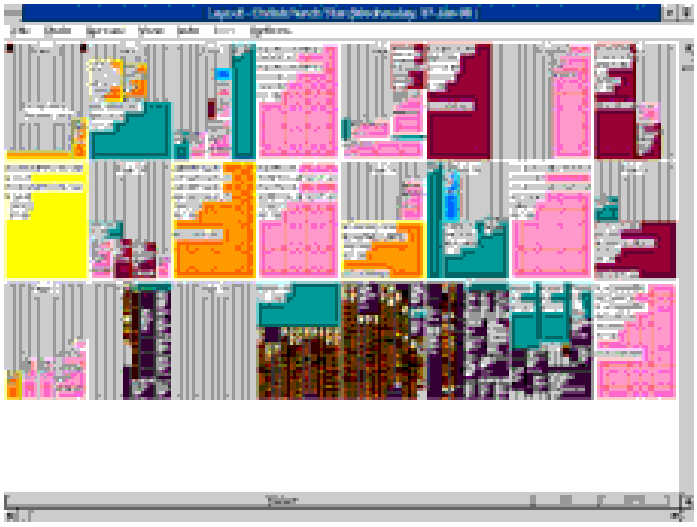

Figura 50: Interfase de usuario de NP Solution

ADS^{NG}/PAGE de Amdocs

http://www.amdocs.com

Los sistemas de preimpresión de Amdocs proporcionan una solución automática e integrada para todas las necesidades operativas en la fase de preimpresión para la producción de directorios y guías telefónicas. Estos sistemas cubren todos los aspectos de la producción de directorios y la publicidad, a través de una gama de prestaciones que agilizan el ciclo de producción, reducen los costos y el tiempo previo a la comercialización. Estos sistemas de preimpresión aplican una tecnología avanzada, que incluye la posibilidad de interacción con sistemas CTP (producción directa a plancha), de modo que el editor puede crear las planchas de impresión directamente desde PostScript. Además, se puede implementar una solución OPI (Open Pre-Press Interface) para producir imágenes de previsualización a baja resolución a partir de imágenes de alta resolución en un entorno PostScript.

ADS^{NG}/PAGE es un sistema automático de paginación en una pasada para la producción de guías telefónicas y páginas amarillas que reduce el proceso de planificación de páginas, a la vez que reduce espacios de relleno, el número de páginas y los costos de impresión, cumpliendo con las reglas de paginación de la editorial.

Este sistema permite agregar encabezados y pies de página, sitúa los listados en columnas y anuncios en cada página. El sistema también automatiza tareas de preparación y finalización de páginas. Además, permite que el planificador de las páginas intervenga en el proceso e introduzca en línea los cambios necesarios. En casos en los que se realizan cambios de último momento en páginas previamente planeadas, el sistema minimiza la acumulación de la carga de trabajo al final del proceso, con lo que se evita que el proceso de producción se vea perturbado.