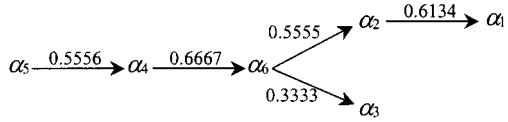


Fig. 7. Minimum spanning tree corresponds to Table VIII.

## VI. Discussions and Conclusions

The primary contribution of this paper is to present a useful tool to support decision making through the data mining technique and the competence set expansion. From simulation results of the numerical example, we can see that it is possible to help decision makers to confidently solve decision problems using the data mining technique and the competence set expansion. Significantly, this is a starting point for integrating data mining techniques with the expansion of the competence set. Each data mining technique, such as clustering, can discover its particular type of useful patterns, which can also be viewed as a competence set for solving one decision problem. Then, the optimal expansion of the competence set with minimum learning cost becomes quite important. In this paper, we first use the proposed data mining technique to find the necessary patterns from a database. Then, we use the minimum spanning table method to optimally expand a needed competence set. For the combination of various techniques of data mining and the competence set expansion, we will study the feasibility and effectiveness.

In fact, the meaning of the linguistic values of quantitative attribute $x_m$ can be changed by a linguistic hedge [9], [10], such as "very" or "more or less." For example

$$\text{very} A^{x_m}_{K,i_m} = \left(A^{x_m}_{K,i_m}\right)^2 \tag{13}$$

$$\text{more or less } A^{x_m}_{K,i_m} = \left(A^{x_m}_{K,i_m}\right)^{1/2}. \tag{14}$$

The membership functions of $(A^{x_m}_{K,i_m})^2$ and $(A^{x_m}_{K,i_m})^{1/2}$ are $[\mu^{x_m}_{K,i_m}(x)]^2$ and $[\mu^{x_m}_{K,i_m}(x)]^{1/2}$, respectively. The use of the linguistic hedge will make the frequent fuzzy grids discovered from a database more friendly and more flexible for decision makers. However, the number of linguistic values of each attribute may be available from domain experts.

## Acknowledgment

The authors are very grateful to the anonymous referees for their valuable comments and constructive suggestions.

## References

[1] M. Berry and G. Linoff, *Data Mining Techniques: For Marketing, Sales, and Customer Support*. New York: Wiley, 1997.
[2] S. Myra, "Web usage mining for web site evaluation," *Commun. ACM*, vol. 43, no. 8, pp. 127–134, 2000.
[3] P. L. Yu and D. Zhang, "A foundation for competence set analysis," *Math. Social Sci.*, vol. 20, pp. 251–299, 1990.
[4] M. J. Hwang, C. I. Chiang, I. C. Chiu, and G. H. Tzeng, "Multistages optimal expansion of competence sets in fuzzy environment," *Int. J. Fuzzy Syst.*, vol. 3, no. 3, pp. 486–492, 2001.
[5] Y. C. Hu, R. S. Chen, and G. H. Tzeng, Discovering fuzzy association rules using fuzzy partition methods, in Knowl.-Based Syst.. (accepted).
[6] J. W. Feng and P. L. Yu, "Minimum spanning table and optimal expansion of competence set," *J. Optim. Theory Appl.*, vol. 99, no. 3, pp. 655–679, 1998.
[7] L. A. Zadeh, "Fuzzy sets," *Inf. Control.*, vol. 8, no. 3, pp. 338–353, 1965.
[8] ——, "The concept of a linguistic variable and its application to approximate reasoning," *Inf. Sci.*, pt. 1, vol. 8, no. 3, pp. 199–249, 1975. (part 2) vol. 8, no. 4, pp. 301–357, 1975; (part 3) vol. 9, no. 1, pp. 43–80, 1976.
[9] H. J. Zimmermann, *Fuzzy Set Theory and Its Applications*. Norwell, MA: Kluwer, 1991.
[10] S. M. Chen and W. T. Jong, "Fuzzy query translation for relational database systems," *IEEE Trans. Syst., Man, Cybern. B*, vol. 27, pp. 714–721, Aug. 1997.
[11] J. W. Han and M. Kamber, *Data Mining: Concepts and Techniques*. San Mateo, CA: Morgan Kaufmann, 2001.
[12] H. Ishibuchi, K. Nozaki, N. Yamamoto, and H. Tanaka, "Selecting fuzzy if-then rules for classification problems using genetic algorithms," *IEEE Trans. Fuzzy Syst.*, vol. 3, pp. 260–270, Aug. 1995.
[13] H. Ishibuchi, T. Nakashima, and T. Murata, "Performance evaluation of fuzzy classifier systems for multidimensional pattern classification problems," *IEEE Trans. Syst., Man, Cybern. B*, vol. 29, pp. 601–618, Oct. 1999.
[14] Y. C. Hu, R. S. Chen, and G. H. Tzeng, "Mining fuzzy association rules for classification problems," *Comput. Ind. Eng.*, to be published.
[15] L. X. Wang and J. M. Mendel, "Generating fuzzy rules by learning from examples," *IEEE Trans. Syst., Man, Cybern.*, vol. 22, no. 6, pp. 1414–1427, 1992.
[16] C. T. Sun, "Rule-base structure identification in an adaptive-network-based fuzzy inference system," *IEEE Trans. Fuzzy Syst.*, vol. 2, no. 1, pp. 64–73, 1994.
[17] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. New York: Plenum, 1981.
[18] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo, "Fast discovery of association rules," in *Advances in Knowledge Discovery and Data Mining*, U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, Eds: AAAI Press, 1995, pp. 307–328.
[19] H. L. Li and P. L. Yu, "Optimal competence set expansion using deduction graph," *J. Optim. Theory Appl.*, vol. 80, no. 1, pp. 75–91, 1994.
[20] Y. C. Hu, G. H. Tzeng, and R. S. Chen, "Discovering fuzzy concepts for expanding competence set," in *Proc. 2nd Int. Symp. Advanced Intelligent Systems*, Daejeon, Korea, 2001, pp. 396–401.
[21] W. Pedrycz and F. Gomide, *An Introduction to Fuzzy Sets: Analysis and Design*. Cambridge, MA: MIT Press, 1998.
[22] P. Adriaans and D. Zantinge, *Data Mining*. Reading, MA: Addison-Wesley, 1996.
[23] J. M. Li, C. I. Chiang, and P. L. Yu, "Optimal multiple stage expansion of competence set," *Eur. J. Oper. Res.*, vol. 120, no. 3, pp. 511–524, 2000.
[24] P. L. Yu, *Forming Winning Strategies: An Integrated Theory of Habitual Domains*. New York: Springer-Verlag, 1990.

# Web Newspaper Layout Optimization Using Simulated Annealing

Jesús González, Ignacio Rojas, Héctor Pomares, Moisés Salmerón, and Juan Julián Merelo

***Abstract*—The web newspaper pagination problem consists of optimizing the layout of a set of articles extracted from several web newspapers and sending it to the user as the result of a previous query. This layout should be organized in columns, as in real newspapers, and should be adapted to the client web browser configuration in real time. This paper presents an approach to the problem based on simulated annealing (SA) that solves the problem on-line, adapts itself to the client's computer configuration, and supports articles with different widths.**

***Index Terms*—Greedy algorithm, pagination, real-time optimization.**

## I. Introduction

Since the amount of information available on the Internet is growing day by day, when a user sends a query to a news site, a lot of information

Manuscript received March 12, 2000; revised January 8, 2002. This work was supported in part by the Spanish CICYT Project DPI2001-3219. This paper was recommended by Associate Editor C. Hsu.

The authors are with the Department of Computer Architecture and Computer Technology, E.T.S. Ingeniería Informática, University of Granada, E-18071 Granada, Spain.

Publisher Item Identifier S 1083-4419(02)03554-9.

is returned and the user frequently has great difficulty in finding the piece of information he really wants, because of the vertical layout most web pages favor. In the case of web newspapers it would be desirable for the final look of the resulting information to be as similar as possible to that of real newspapers. The layout of the articles should be organized in columns without any overlap so the user can read all of them, occupy the smallest possible area, minimize gaps between articles, and avoid scroll bars if possible to make reading easier for the user.

As the query is sent via a web browser, the results should also be presented as a web page and as soon as possible. This requirement transforms the layout optimization problem into an on-line problem which has to be solved in real time, so a fast algorithm is needed to solve the problem quickly. The algorithm proposed in this paper solves the problem rapidly compared with the time spent loading the web page to be optimized, while taking into account the font sizes and faces the user is using and the size of the web browser window where the result is to be displayed, thus obtaining a result that is adapted to the client's computer configuration.

One approach to this problem is to do all the optimization on the server side, just when the user's query is received. This approach is used in the *YPPS++* system in Germany [3] to paginate Yellow Pages. This layout optimization problem does not need to be solved in real time, so it is done off-line and the Yellow Pages are printed later.

The above approach is also used in [11] to optimize fax newspapers and Yellow Pages in several countries. In this paper, three different methods are presented, a heuristic method and two other ones that use simulated annealing (SA). The methods using SA obtain better results, but overlapping between articles is allowed and even with the best of the SA methods a slight overlap of articles in the final result is sometimes observed.

Applying this approach to the web newspaper optimization problem has two main disadvantages. The first one is that we would obtain a generic result that might not be well suited to the user's web browser configuration, and the second and most important one, that if the news server has lots of queries to solve at the same time, it will become overloaded.

If the results must be ready in real time, a different approach, in which the optimization is carried out in the client side, is needed. If the algorithm that performs the layout optimization is sent in the same web page to be optimized as a script that will be interpreted in the web browser when the page loading finishes [5], [6], the server only has to look for the articles that match the user's query and send them, this being much faster than in the above approach. The optimization takes place in the client's computer web browser, just after the web page containing the articles and the optimization algorithm is loaded, taking into account the web browser window dimensions where the information must be displayed and the font faces and sizes being used by the user, thus obtaining a result that is adaptable to every possible client web browser configuration without overloading the server.

In the field of on-line approaches to the pagination problem there are some examples like the *Krakatoa Project* [8], which implements a personalized newspaper as a *Java* Applet embedded in a web page and is able to customize the final layout depending on the user profile, although it does not optimize the total surface occupied.

Another approach is described in [5], where the articles extracted from several newspapers as the result of a user query are displayed on a web page that attempts to provide a newspaper-like layout. This approach is based on a genetic algorithm (GA) where the top-left corner of each article is encoded as a gene in every chromosome in the population. This representation is not very good because it requires a lot of calculation to check whether two or more articles are overlapping, making the algorithm very slow.

Another approach based on SA [6] use a different representation that obtains better results ten times more quickly than in [5], but with the constraint of all articles having the same fixed width. The surface of the web browser window is divided into several columns of a fixed width and all the articles must have the same width as the columns. Possible solutions are represented as permutations of articles to be displayed using a greedy algorithm that allocates the articles in the least occupied column following the order they have in the permutation. This approach is much faster than the one in [5] because the representation it uses does not allow gaps or article overlapping, but suffers the restriction that all the articles must be of the same width, while in [5] several article widths are allowed.

Usually, visual aesthetics do not play a part in the optimization of newspaper layout, although there have been some failed attempts to optimize layout aesthetics reported in [1]. Usability and aesthetics for websites, is, anyway, a complex problem, it is difficult to evaluate it numerically, and thus, difficult to optimize; it is more an art than a science, as expressed by Jakob Nielsen in his website UseIt (`http://www.useit.com`). That is why, in this paper, we are not making any claim of usability or aesthetics optimization, other than the fact that concentrating information without leaving gaps avoids scrolling, making headlines of more articles accessible at once.

This paper describes an SA-based algorithm for the web pagination problem that optimizes the layout of all the articles matching the client's query to a web newspaper in real time and makes all the optimization task in the client computer. The algorithm is able to manage articles with several widths and generates a web page with an appearance as similar as possible to a real newspaper, taking into account the size of the browser window and the face and size of the fonts in the client's machine, and producing a layout that adapts itself to the user's computer characteristics.

The layout of this paper is as follows. The proposed approach is discussed in Section II, the results obtained are analyzed in Section III, and some conclusions are drawn in Section IV.

## II. Proposed Approach

In real newspapers, pages are divided into several columns; and articles, depending on their length, can occupy one or more columns. Following this idea, the web browser window is divided into several columns of fixed width, fitting in the available surface and avoiding horizontal scroll bars, because it is annoying to scroll the page to the right to finish reading one line and later to scroll the page to the left to begin reading the following one.

As the number of articles to be returned by the server is unknown *a priori*, the columns of the page have infinite length, and a vertical scroll bar would appear if there were not sufficient room for all the articles in the available surface.

All articles to be displayed have a width that is a multiple of the width of one column, depending on their length. The problem to be solved is how to fill all the columns of the web page equally, minimizing gaps between articles. This problem is apparently similar to a two-dimensional bin packing problem [12] if columns are identified with bins, but in a bin packing problem the goal is to minimize the amount of bins required to pack several objects, while in this problem, the number of columns (bins) is fixed *a priori* and what has to be minimized is the unused capacity of the bins (columns).

This representation was first used in [6], but in that approach, articles were restricted to fit in only one column, so it was impossible to generate gaps between articles and the fitness function only had to compare the capacity used of the most filled column with the capacity used of the least filled column to estimate the unused capacity. Since in the approach presented in this paper articles may have different widths, it is possible to generate gaps between articles (see Figs. 1 and 2), so the fitness function and the greedy algorithm to allocate the articles in the web page have been changed to support this new feature and to try to minimize the size of the gaps between articles.

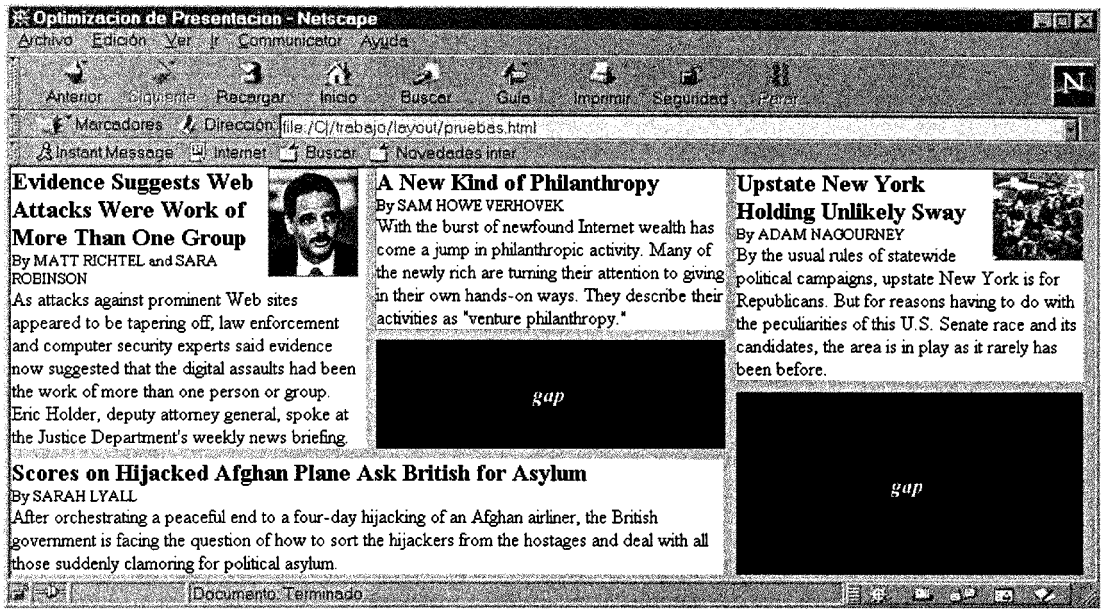


Fig. 1. Bad allocating of articles.

There are several paradigms to face optimization problems like the one presented in this paper. Between the most popular, we have SA [2], [10] and GAs [4], [7], [13].

SA has been widely used to solve combinatorial problems. It is inspired by the physical process of heating a substance and then cooling it slowly until a strong crystalline structure is obtained. This process is simulated by lowering an initial temperature by slow stages until the system "freezes" and no more changes occur. Each stage in the process consists in changing the configuration several times until a *thermal equilibrium* is reached and a new stage can start with a lower temperature; the configuration is obtained in the last stage. The changes in the configuration are performed in the following way: a new configuration is built by a random displacement of the current one. If the new configuration is better, then it replaces the current one, and if not, it may replace the current one probabilistically. The probability of replacing the current configuration by a worse one is high at the beginning of the process and decreases in every stage. This procedure allows the system to move consistently toward the best configuration, yet still enables escape from local optima due to the probabilistic acceptance of worse configurations during the optimization task. However, SA is not guaranteed to find the global optima; it is just a bit better than other algorithms escaping from local optima; the solution obtained by SA can be called a "good enough" solution, but it is not guaranteed to be the best.

As SA only evolves one potential solution instead of a whole population like GAs, it is much faster, and as the web newspaper optimization problem must be solved in real time, SA is more appropriate than GAs to find a good solution for the problem quickly.

The most important parts in an SA algorithm are: the objective function to be minimized during the optimization process, the chosen representation for potential solutions to the problem and the mutation or configuration change operator. These three characteristics are presented in the next subsections.

### A. Objective Function

The objective function estimates the cost of a layout by measuring the unused surface or amount of gaps in it. Gaps may appear in a layout on attempting to allocate an article that is wider than a column as is the case of the gap in the second column in Fig. 1 or when there are no more articles to place and there is unused space in some columns as is the case of the gap in the third column in Fig. 1 and the gaps in Fig. 2.

The objective function to be minimized is just the mean height of all the gaps existing in the layout

$$f = \frac{\sum_{j=0}^{n} gap_j}{n} \tag{1}$$

where $n$ is the number of gaps generated in the layout.

By minimizing this function it is possible to obtain a layout with several gaps, but which are very small, almost imperceptible in most cases.

### B. Problem Representation

Every possible solution has been coded as a permutation of the articles to be displayed. To obtain the layout encoded in the permutation, the system uses a decoder which allocates the articles following the order imposed by the permutation. Each different permutation will produce a different layout. For example, with nine articles to be laid out, and the following permutation:

$$solution = (\,4 \quad 6 \quad 1 \quad 9 \quad 2 \quad 7 \quad 5 \quad 3 \quad 8\,)$$

the article number 4 will be allocated in the first place, followed by the article number 6, and by the remaining ones, allocating the article number 8 after all the other ones. With this representation the problem can be seen as a combinatorial problem of size $N$, being $N$ the number of articles to be laid out.

The decoder uses a greedy algorithm that allocates all the articles following the order imposed by the permutation encoded in the current configuration, and for every article to place, it calculates the mean of generated gap sizes generated above the article and the dispersion of these gap sizes, $m_i$ and $d_i$, respectively, using (2) and (3) for each column $i$ where it is possible to allocate the article, choosing the column with the smallest dispersion $d_i$. If there is more than one column with minimal dispersion, the minimal mean $m_i$ is used to choose the column, and if there is more than one column with equal dispersion and mean values, it chooses the leftmost one (the one with smallest $i$)

$$m_i = \frac{\sum_{j=0}^{n_i} gap_j}{n_i} \tag{2}$$

$$d_i = \frac{\sum_{j=0}^{n_i} |gap_j - m_i|}{n_i} \tag{3}$$

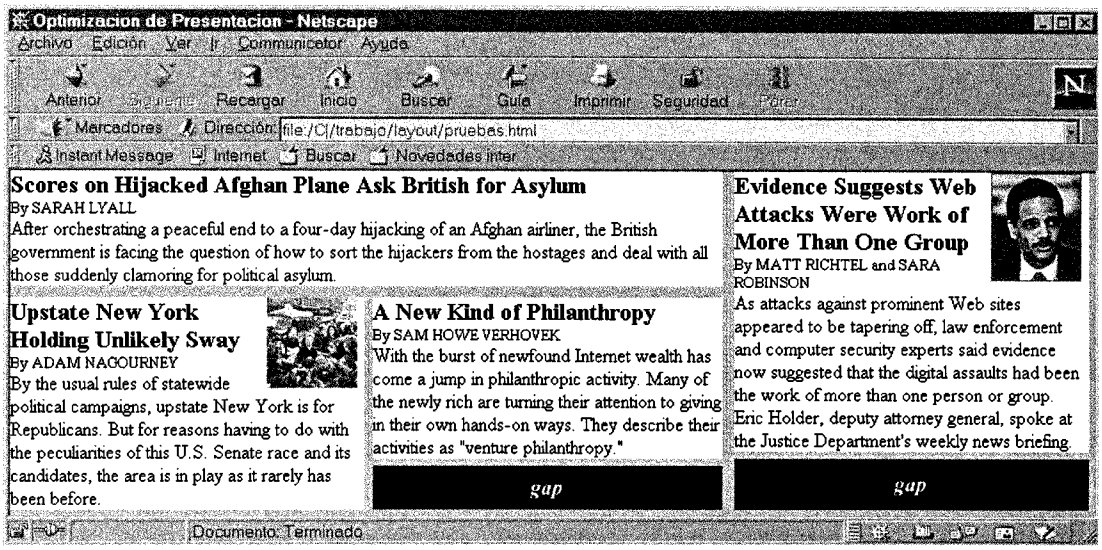


Fig. 2. Good allocating of articles.

TABLE I
MINIMUM, MAXIMUM, AVERAGE, AND STANDARD DEVIATION OF TIME AND COST OPTIMIZING 10, 25, 50, AND 100 ARTICLES

| ARTICLES | $T_{min}$ | $T_{max}$ | $T \pm \sigma$ | $C_{min}$ | $C_{max}$ | $C \pm \sigma$ |
|---|---|---|---|---|---|---|
| 10 | 4.45 | 4.56 | 4.49 ± 0.03 | 1.67 | 3.5 | 2.08 ± 0.58 |
| 25 | 9.68 | 10.9 | 9.95 ± 0.41 | 0 | 7 | 2.76 ± 2.13 |
| 50 | 18.38 | 21.96 | 19.43 ± 1.13 | 2 | 12 | 6.7 ± 3.20 |
| 100 | 39.06 | 59.19 | 43.54 ± 3.20 | 5.14 | 13.64 | 9.03 ± 2.81 |

where $n_i$ is the number of gaps generated above the article when allocating it in column $i$ and $gap_j$ is the height of the $j$th gap generated.

Allocating the articles in this way produces layouts where there might be many gaps, but with a very small size, almost imperceptible in most cases, due to the objective function used, see (1), while trying to minimize the number of gaps would result in a layout with fewer but bigger gaps, which is worse from the aesthetic point of view (see Figs. 1 and 2).

### C. Mutation Operator

To obtain a new configuration randomly in the neighborhood of the current one, a transposition of two articles is performed in the following way. For a configuration, two articles are selected randomly (the underlined ones)

$$old = (1 \quad 2 \quad \underline{3} \quad 4 \quad 5 \quad 6 \quad 7 \quad \underline{8} \quad 9).$$

The new permutation generated is a copy of the old one, but with the numbers at the marked positions swapped

$$new = (1 \quad 2 \quad \underline{8} \quad 4 \quad 5 \quad 6 \quad 7 \quad \underline{3} \quad 9).$$

### D. Algorithm

The algorithm that performs the optimization is detailed in the following code:

```
v = 0
T = T0
select a configuration c_cur at random
evaluate c_cur
repeat
   for j = 1 to k
      select a new configuration c_new in
  the neighborhood of c_cur by mutating c_cur
      Δf = f(c_new) − f(c_cur)
      if (Δf < 0) OR (random(0, 1)
  < e^(−(Δf/T)))
      then c_cur = c_new
   endfor
   T = f_T(T0, v)
   v = v + 1
until (T < T_min)
Use the last configuration to obtain the
  layout
```

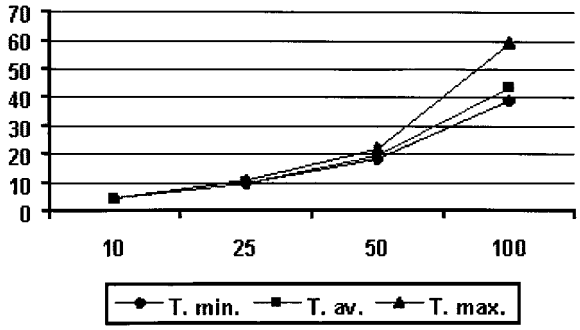


Fig. 3. Minimum, average, and maximum time to optimize 10, 25, 50, and 100 articles.

where $v$ counts the number of iterations performed; $T$ keeps the current temperature; $T_0$ is the initial temperature and $T_{\min}$ is the minimum temperature to reach; $c_{cur}$ keeps the current configuration, $c_{new}$ is the

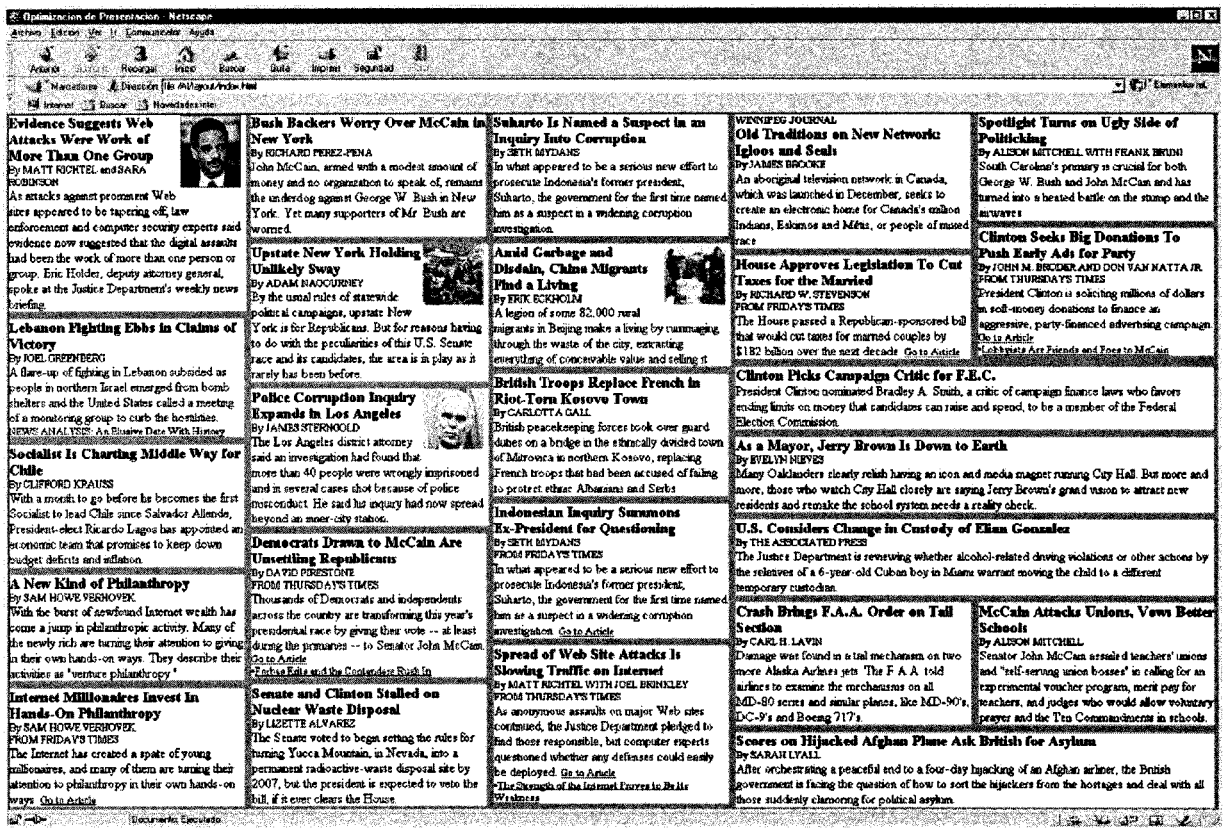


Fig. 4. Final result of a simulated newspaper page with 25 articles.

new configuration selected in the neighborhood of $c_{cur}$; $f$ is the objective function; $k$ is the number of changes to reach the thermal equilibrium; and $f_T$ is the freezer function.

The initial temperature is calculated following Kirkpatrick's suggestion [9]

$$T_0 = -\frac{\Delta f^*}{\ln(p_a)} \tag{4}$$

where $\Delta f^*$ is the average objective increase observed in a random change, and $p_a$ is the initial acceptance probability (0.8 is usually used).

For the freezer function ($f_T$) this approach uses

$$f_T(T_0,\, v) = \frac{T_0}{1+v}. \tag{5}$$

This function lowers the temperature and thus the acceptance probability of a solution worse than the current one. Initially the temperature is high, producing an exploration of the search space, and later starts a controlled descent where the algorithm searches locally until the minimum temperature is reached.

The minimum temperature is calculated on the basis of the desired number of iterations $numIt$ as follows:

$$T_{\min} = f_T(T_0,\, numIt). \tag{6}$$

## III. Results

The algorithm has been tested with several articles extracted from the web newspaper *New York Times* (`http://www.nytimes.com`).

Table I shows the minimum, maximum, average, and standard deviation of time ($T$) in seconds and cost (unused surface) ($C$) in pixels measured over ten runs for various numbers of articles executed within Netscape Communicator 4.51 running in a 233 MHz Intel Pentium II. The parameters used in all the algorithm runs were $numIt = 80$ and $k = 10$.

One interesting feature of this algorithm is that the average search time increases almost linearly with the size of the problem, as shown in Fig. 3, where the minimum, average, and maximum times in Table I are plotted.

It is important to note that the problem to be solved in this paper is more difficult than the one reported in [6], because as articles can span more than one column, gaps that might appear between articles have to be minimized. If the number of iterations is increased, the algorithm presented in this paper is able to find better solutions, but as the number of articles increases, so does the search time, becoming very slow at finding the optimal solution; while in 80 iterations, it finds an appropriate solution to the problem in a reasonable time.

Since the algorithm is written in *JavaScript* [14], it must be interpreted within the web browser, which is a slower process than running a compiled program. Even under this restriction, it obtains reasonable times when optimizing fewer than 50 articles compared with the time spent while the web page is loaded, and taking into account that in a normal-sized browser window with a standard (readable) font size (i.e., 10–12 points), there is only room for ten articles without scrolling bars in the window and that the usual number of articles returned from the server is no greater than 25, optimization times are usually between 4.5 s(with ten articles) and 10 s (with 25 articles), which is acceptable for a personalized optimization of the layout.

Another interesting feature of this algorithm is the quality of the obtained results. As stated in Section II, SA itself does not guarantee global convergence and hence cannot assure global optimality, but if we take a look at Table I, we can observe that the cost (the mean height of the gaps) of the obtained results is about two or three pixels for a usual web page containing 10–25 articles. Table I also shows that the standard deviation over the mean costs is as much as another two pixels. These results give an insight of how close the obtained layouts are from the optimum one. As an example, Fig. 4 shows a final layout where 25 articles are displayed using a very small font size to obtain a layout that

fits inside a window without scrolling bars. "Eyeball" examination of this figure shows that the existing gaps are almost indistinguishable.

## IV. Conclusions and Future Work

This paper presents a new approach based on SA to the web newspaper layout problem. It adds the feature of being able to deal with articles with different widths to the one presented in [6]. As in [6], the optimization program is a script written in *JavaScript* embedded in the web page to be optimized, which will be executed in the client's machine, adapting itself to the client web browser configuration.

The time required for the optimization process is acceptable compared with the usual time the user has to wait while the web page loads; e.g., in a 233 MHz Intel Pentium II it is usually between 4.5 s (with ten articles) and 10 s (with 25 articles); with a Pentium II 450 MHz it barely takes 1 s. From this site, downloading an offsite page takes around 1.5 s, so that the total (downloading + optimizing) would be less than 5 s; that is, an increment barely noticeable over the usual downloading time. As processor speed improves, the optimization time should be better, so this time is a very good one, taking into account that the optimization task is executed by the *JavaScript* engine in the browser.

The algorithm proposed in this paper is available in `http://atc.ugr.es/~jesus/layout` for anyone interested in executing or taking a look at the algorithm.

In the future, one of the lines of research will go along improving usability and aesthetics together with the layout, as well as taking into account news item "weight," so that "heavier" news items should preferably go to the top of the page, while less important news should go down the page.

## References


[1] A. Aardal. (1996) WORK PACKAGE 4.3, Logistics and resource management, ALCOM - IT, Algorithms and complexity in information technology. ESPRIT Project 20244. [Online]. Available: http://www.cs.ruu.nl/research/projects/alcom/wp4.3.html.

[2] E. H. L. Aarts and J. Korst, *Simulated Annealing and Boltzmann Machines*. New York: Wiley, 1989.

[3] German Research Center for artificial intelligence GmbH. Yellow-Pages Pagination System. [Online]. Available: http://www.dfki.de/imedia/ypps/index.html.

[4] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[5] J. González and J. J. Merelo, "Optimizing web page layout using an annealed genetic algorithm as client-side script," in *Proceedings of the 5th Conference on Parallel Problem Solving From Nature—PPSN IV*, A. E. Eiben, T. Baeck, M. Schoenauer, and H. P. Schwefel, Eds. New York: Springer-Verlag, Sept. 1998, vol. 1498, Lecture Notes in Computer Science, pp. 1018–1027.

[6] J. González, J. J. Merelo, P. A. Castillo, V. Rivas, and G. Romero, "Optimizing web newspaper layout using simulated annealing," in *International Work-Conference on Artificial and Natural Neural Networks, IWANN'99, Vol. II*, J. Mira and J. V. Sánchez-Andrés, Eds. New York: Springer-Verlag, June 1999, vol. 1607, Lecture Notes in Computer Science, pp. 759–768.

[7] J. J. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: Univ. of Michigan Press, 1975.

[8] O. Kamba, K. Bharat, and M. C. Albers, "The Krakatoa Chronicle—An interactive, personalized newspaper on the web," Graphics, Visualization and Usability Center, Georgia Institute of Technology, Atlanta, Tech. Rep. 95-25, 1995.

[9] S. Kirkpatrick, "Optimization by simulated annealing—Quantitative studies," *J. Stat. Phys.*, vol. 34, pp. 975–986, 1984.

[10] S. Kirkpatrick, C. D. Gerlatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–680, 1983.

[11] K. Lagus, I. Karanta, and J. Ylā̄a-Jā̄aski, "Paginating the generalized newspapers—A comparison of simulated annealing and a heuristic method," in *Proceedings of the 4th Conference on Parallel Problem Solving From Nature—PPSN IV*, H. M. Voigt, W. Ebeling, I. Rechenberg, and H. P. Schwefel, Eds. New York: Springer-Verlag, Sept. 1996, vol. 1141, Lecture Notes in Computer Science, pp. 595–603.

[12] S. Martello and P. Toth, "Knapsack problems, algorithms and computer implementations," in *Bin Packing Problem*. New York: Wiley, 1990, ch. 8.

[13] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd ed. New York: Springer-Verlag, 1996.

[14] Netscape Communications Corporation. JavaScript developer central. [Online]. Available: http://developer.netscape.com/tech/javascript.


# Hierarchical Semi-Numeric Method for Pairwise Fuzzy Group Decision Making

M. Marimin, Motohide Umano, Itsuo Hatono, and Hiroyuki Tamura

***Abstract*—Gradual improvements to a single-level semi-numeric method, i.e., linguistic labels preference representation by fuzzy sets computation for pairwise fuzzy group decision making are summarized. The method is extended to solve multiple criteria hierarchical structure pairwise fuzzy group decision-making problems. The problems are hierarchically structured into focus, criteria, and alternatives. Decision makers express their evaluations of criteria and alternatives based on each criterion by using linguistic labels. The labels are converted into and processed in triangular fuzzy numbers (TFNs). Evaluations of criteria yield relative criteria weights. Evaluations of the alternatives, based on each criterion, yield a degree of preference for each alternative or a degree of satisfaction for each preference value. By using a *neat* ordered weighted average (OWA) or a fuzzy weighted average operator, solutions obtained based on each criterion are aggregated into final solutions. The hierarchical semi-numeric method is suitable for solving a larger and more complex pairwise fuzzy group decision-making problem. The proposed method has been verified and applied to solve some real cases and is compared to Saaty's analytic hierarchy process (AHP) method.**

## I. Introduction

In pairwise fuzzy group decision making, we have a set of $n$ alternatives $S = \{s_1, s_2, \ldots, s_n\}$, and a set of $m$ decision makers $I = \{1, 2, \ldots, m\}$. Each decision maker $k \in I$ provides his or her fuzzy preference over $S$. A solution is an alternative (or a set of alternatives) which is the most acceptable to a group of decision makers as a whole. The decision-maker preferences can be expressed in single-point numeric values and then processed by using crisp computation models [1], [2]. The crisp computation models have been extended into Nurmi's $\alpha$ degree models [3] which are applied by Kacprzyk [4], Kacprzyk *et al.*

Manuscript received June 25, 1997; revised June 24, 2001 and January 26, 2002. This paper was recommended by Associate Editor L. O. Hall.

M. Marimin is with the Department of Agro-Industrial Technology, Faculty of Agricultural Technology, Bogor Agricultural University, Bogor 16002, Indonesia.

M. Umano is with Department of Mathematics and Information Sciences, College of Integrated Arts and Sciences, Osaka Prefecture University, Osaka 593, Japan.

I. Hatono is with Kobe University, Kobe 657–8501, Japan.

H. Tamura is with Department of Systems and Human Science, Graduate School of Engineering Science, Osaka University, Osaka 560-8531, Japan.

Publisher Item Identifier S 1083-4419(02)04372-8.