Revista de Matemática: Teoría y Aplicaciones 2006 **13**(2) : 125–138
cimpa – ucr – ccss issn: 1409-2433

HEURISTIC ANALYSIS OF A NEAR OPTIMAL APPROXIMATION ALGORITHM FOR THE DETERMINATION OF INVESTMENT OPTIONS

Juan Flores* Javier Ávila† Federico González‡
Beatriz Flores§

Recibido/Received: 26/08/04 — Aceptado/Accepted: 22/09/06

Abstract

When cash becomes available in a company, there are several strategies that allow us to benefit from it. The problem is how much to invest, for how long, and using which of the investment options in order to get the maximum profit out of it. A common problem in business administration is that we do not want to keep the money idle in the checking account, neither to over-invest. When the cash function becomes negative an analogous scheme is used as we want to pay as little interests as possible. In this paper we are reporting the experiments and implementation of several heuristics that can be used with the greedy algorithm and how well they behave. Finally we develop a hybrid algorithm that takes the best of the greedy algorithm and performs a very limited search. We find in this work that with the greedy algorithm we use, in general is not possible to optimize the profit for a given function; nevertheless the algorithm we use can find profits that are very close to the optimum and in some cases it gets the optimum. The proposed algorithm use a heuristic search based on the greedy scheme and greedy selection criteria to find profits close to the optimum. A software application was developed in order to show that the proposed strategy really works. Although this algorithm is suboptimal, it is very efficient in terms of time.

Keywords: Common sense reasoning, knowledge representation,soft decision making, greedy algorithms, financial analysis.

*Facultad de Ingeniería Eléctrica. División de Estudios de Posgrado, Universidad Michoacana de San Nicolás de Hidalgo. Morelia, Michoacán, México. E-Mail: juanf@zeus.umich.mx

†Igual que J. Flores. E-Mail: javila@lsc.fie.umich.mx

‡Facultad de Contabilidad y Administración. División de Estudios de Posgrado, Universidad Michoacana de San Nicolás de Hidalgo. Morelia, Michoacán, México. E-Mail: fsantoyo@zeus.umich.mx

§Igual que F. González. E-Mail: bflores@zeus.umich.mx

Resumen

Cuando se tiene disponibilidad de dinero en efectivo en una compañía, existen varias estrategias que nos permiten obtener beneficios de éste. El problema es determinar cuanto invertir, por cuanto tiempo y que opciones de inversión usar para obtener la máxima utilidad. Un problema común en la administración de negocios es que no se quiere mantener el efectivo ocioso en la cuenta de cheques, ni tampoco sobre invertir. Cuando la función de dinero disponible es negativa, se utiliza un esquema análogo, en este caso se busca pagar la menor cantidad de dinero por intereses generados. En este artículo estamos reportando los experimentos e implementación de varias heurísticas que pueden ser usadas con el algoritmo voraz, donde analizamos su desempeño. Finalmente se desarrolla un algoritmo híbrido que toma el mejor resultado de varias heurísticas utilizando el algoritmo voraz y hace una búsqueda muy limitada. En este trabajo encontramos que con el algoritmo voraz utilizado, no es posible optimizar en general la utilidad para un problema dado; sin embargo, el procedimiento utilizado puede encontrar utilidades muy cercanas al óptimo y en ocasiones conseguir el óptimo. Se desarrolló una aplicación computacional para mostrar que la estrategia propuesta realmente funciona. Aunque el algoritmo propuesto es subóptimo, es muy eficiente en tiempo de procesamiento.

Palabras clave: Razonamiento de sentido común, representación de conocimiento, toma de decisiones, algoritmos voraces, análisis financiero.

Mathematics Subject Classification: 00A72, 90C27, 90C90, 68W25

1 Introduction

In the real world there are several problems that are very difficult in its essence because the number of possible solutions fall in the combinatory field; that is the case of the problem we are tackling in this paper. To find the best combination of investment or credit assignment options among all that can occur is not an easy task.

When cash becomes available in a company there is a problem of major interest for managers. A good example of it is to find the best mix of investment assignment options in order to get the maximum profit out of it.

If you are not to invest in real estate or other assets, you may choose a different investment option; fixed interest can be an important part of everyone's asset allocation strategy with low risk where the money you are handling earns some interests through time. The main problem in this case is to find how much to invest, for how long, and using which of the available investment options. You do not want to keep the money in the checking account, because it would not produce any profits; neither do you want to invest too much, because you will need that money to operate your company.

On the other hand, if you run out of money, you need to get a loan to keep your business working until the money becomes available again; that means you have to find the best mix of credit assignment options in order to minimize the cost of the money. In this case, you do not want to borrow more than necessary and pay interests for idle money; neither do you want to borrow less, or you will run out of cash.

Section 2 defines the problem we are assessing. Section 3 presents a recursive solution to the problem and its selection criteria. Section 4 reports the results of the heuristics used with the greedy approach and implementation. Finally, section 5 gives the conclusions and discusses possible extensions to the present work.

2 Problem statement

Our problem starts with a cash availability time series, where $F(t_i)$ represents available cash at time t_i, for $i = 1, 2, ..., n$, which will be called F from now on. F can be obtained using the total amount of cash available at any moment, those quantities result from planning the incomes and expenses that the company will have during the exercise.

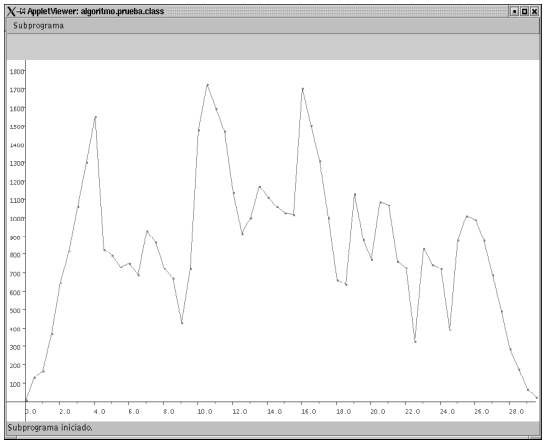

Figure 1: Cash Availability Function F.

Figure 1 represents an example of F that will be used to illustrate the different concepts and algorithms exposed throughout this paper.

As mentioned in the introduction, we do not want the available cash to sit idle, not producing any profits. On the contrary, our aim is to find the best mix of investment assignment options in order to get the maximum profit out of it. Figure 2 shows the investment options used in our example. Each option have a name, a minimum time to invest, a minimum amount of money, and an annual percentage rate APR for every option. The investment options are ordered by monotonically decreasing APR. Also, each option is assigned a color, that will be useful in the visualization of the solution.

When F becomes negative, we have a situation where we need to spend money that is not available. In such cases, we need to use a loan. Similarly to investments, we do not want to run short and use a smaller loan than needed. Neither do you want to use a larger loan and have to pay interest for money you are not going to use. For this particular case, it occurs the opposite in an available loan options scheme, as we borrow more money at a longer time, we will have an annual percentage rate that falls.

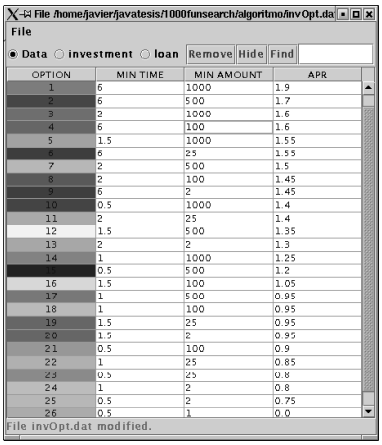

Figure 2: Available Investment Options.

An investment optimization problem, by the above definitions, is represented as a triplet (F, I, L), where F is the cash availability time series as mentioned before, I and L are the available investment and loan options, respectively.

2.1 Investment options assignment

An assignment is a triplet $(inv, interval, amount)$, meaning we are going to invest an *amount* of money for a period of time *interval*, using an investment option *inv*; *interval* is given by the initial and final times of the investment assignment. The investment option *inv* is given by the available investment options AIO where each element of the list has (see figure 2):

- *An investment name* inv.
- *Minimum time to invest* MTI.
- *Minimum amount to invest* MAI.
- *Annual percentage rate* APR.

An assignment A, given by $(inv, interval, amount)$, is considered feasible if it satisfies the following constraints:

$$inv \in AIO \tag{1}$$

$$|interval| \geq MTI \tag{2}$$

$$amount \geq MAI \tag{3}$$

where

$$|interval| = t_{final} - t_{initial} \tag{4}$$

$$amount = min_{t \in interval}(F(t)) \tag{5}$$

An assignment A is represented by a rectangular area made up of *amount* (height) and *interval* (width); each assignment has an investment option *inv* and a specific annual percentage rate APR, associated to option *inv*.
The value of a rectangular object i is the profit or return that our investment yields and is given by (6):

$$value_i = |interval_i| amount_i APR_i \frac{1}{(12)(100)} \tag{6}$$

We define the weight of each object as in (7)

$$weight_i = |interval_i| amount_i \tag{7}$$

2.2 Loan options assignment

Each assignment A is represented by an *interval* that is the loan time, an *amount* of cash we need to borrow using a loan option *loan*. An assignment is thus a triplet $(loan, interval, amount)$, meaning we are going to borrow an *amount* of money for a period of time *interval*, using a loan option *loan*. *Interval* is given by an initial and final time of the loan assignment. The loan option *loan* is given by the available loan options ALO where each element of the list has:

- *A loan name loan.*
- *Minimum time to borrow* MTL.
- *Minimum amount for the loan* MAL.
- *Annual percentage rate* APR.

An assignment A given by $(loan, interval, amount)$ is considered feasible if it satisfies the following constraints:

$$loan \in ALO \tag{8}$$

$$|interval| \geq MTL \tag{9}$$

$$amount \geq MAL \tag{10}$$

where

$$|interval| = t_{final} - t_{initial} \tag{11}$$

$$amount = min(F(t_{initial}), F(t_{final})) \tag{12}$$

An assignment A is represented by a rectangular area made up of *amount* (height) and *interval* (width), each assignment has a loan option *loan* and a specific annual percentage rate APR.
The value of a rectangular object i is the cost of the loan and is given by (13):

$$value_i = |interval_i|(-amount_i) APR_i \frac{1}{(12)(100)} \tag{13}$$

We define the weight of each object as in (14)

$$weight_i = |interval_i| amount_i \tag{14}$$

2.3 Objective function and constraints

Our objective is to find the possible assignments set that maximize the profit of investments, and minimize the cost of the credits.

For an investment scheme, the problem could be seen as a total area under a discrete curve that is equals to the total weight W, to be filled by rectangular objects O_i for $i = 1, 2, ..., n$. The i^{th} object should have a positive $weight_i$ and a positive $value_i$ where the total sum of its values be maximized and the total sum of its weights be less than or equal to W.

$$maximize \sum_{i=1}^{n} value_i \tag{15}$$

For a loan scheme, where F becomes negative, everything remains the same but the total sum of its values should be minimized.

$$minimize \sum_{i=1}^{n} value_i \tag{16}$$

(15) and (16) are subject to the following constraint

$$\sum_{i=1}^{n} weight_i \leq W \tag{17}$$

where $value_i > 0$, $weight_i > 0$ for $1 \leq i \leq n$

3 A recursive solution

We have an optimization problem, is a computational problem where the goal is to find the best of all possible solutions. More formally, we need to find a solution in the search space defined by the problem which maximizes Eq. 15 or minimizes Eq. 16.

The available investment or credit options yield a value given by Eqs. 6 and 13 respectively, to the different money assignments in a given period of time. These assignments could be seen as rectangular objects that fit under the curve F with a specific value and weight defined in Eqs. 7 and 14.

Next, we give the definition of a basic greedy assignment:

$$A = (F[t_i..t_f], I, L) \tag{18}$$

which takes as a parameter an array $F[t_i..t_f]$ containing a sequence of length n that is to be evaluated, where t_i is the initial time, t_f is the final time and represents the available cash for a given problem, I is an assignment for investments and L is an assignment

for loans; an assignment is thus a triplet $(inv|loan, interval, amount)$ as defined in the previous section; given the basic greedy assignment the three subproblems are defined as follows:

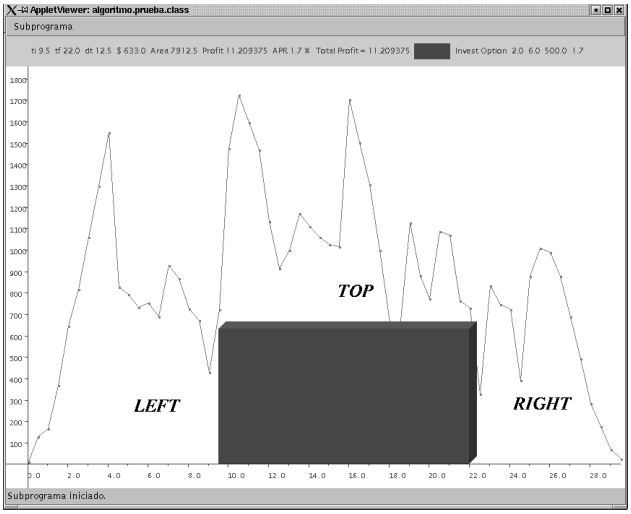

Figure 3: Plot of function F with the first basic assignment.

$$LEFT = (F[t \leq t_i], I, L) \tag{19}$$

$$TOP = (F[t_i..t_f] - amount, I, L) \tag{20}$$

$$RIGHT = (F[t \geq t_f], I, L) \tag{21}$$

Given the above definitions and those in the preceding section, the problem reduces to determine a set of assignments that patch the area under the function F. We can start the patching by determining a basic greedy assignment. Once this basic greedy assignment is selected by one of the selection criteria listed in the following section, the problem reduces to three similar subproblems of lower complexity. The subproblems are identified as $LEFT$, $RIGHT$ and TOP. Figure 3 shows the first basic assignment for our example.

The definitions given in Eqs. 18, 19, 20 and 21 are basis for the algorithm GIA proposed by *Flores et al.* in [3].

Algorithm GIA (Greedy Investment Assignment) takes a problem and returns a set of investment (or loan) assignments. Algorithm GIA is the general strategy used to obtain investment or loan assignments, where A is the *basic assignment* and $LEFT$, TOP and $RIGHT$ contain the assignments that provide solutions for their respective subproblems. $LEFT$, TOP and $RIGHT$ subproblems are solved recursively, and then their solutions combined to solve the original problem (see figure 4).

Figure 5 shows a complete solution for the example problem corresponding to the best solution found by our procedure.

$GIA(F, I, L)$
if$(F \neq \varnothing)$ **then**
$A = basic\ assignment(F[t_i..t_f], I, L)$
$LEFT = GIA(F[t \leq t_i], I, L)$
$TOP = GIA(F[t_i..t_f] - amount, I, L)$
$RIGHT = GIA(F[t \geq t_f, I, L])$
return $A + LEFT + TOP + RIGHT$

Figure 4: Greedy Investment Assignment Algorithm.

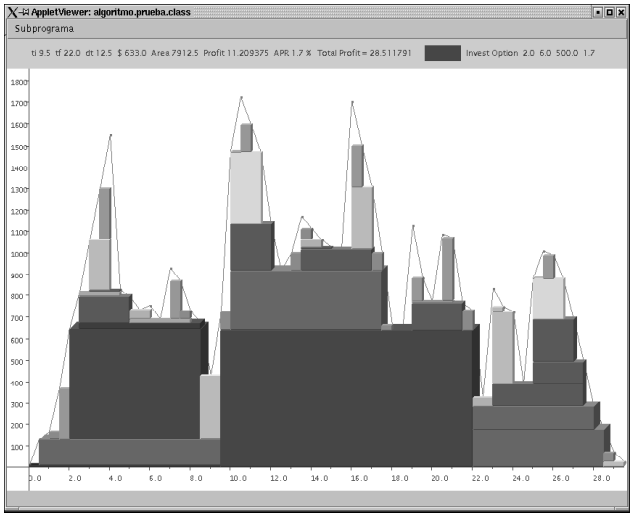

Figure 5: Complete solution for the example problem.

The following subsections mention the greedy strategies used to make the choice from a set of feasible assignments. How do we choose our basic greedy assignment from all possible feasible assignments? Here, the greedy algorithm chooses the basic assignment using one of the selection criteria and then recursively makes the choices in the $LEFT$, $RIGHT$ and TOP subproblems. A greedy algorithm obtains an optimal solution to a problem by making a sequence of choices. For each decision point in the algorithm, the choice that seems best at the moment is chosen [2]. This heuristic strategy does not always produce an optimal solution, but sometimes it does.

In the algorithm proposed in this paper, the choice is done by selection criteria functions to maximize the total sum of all the feasible assignments values for investments or minimize the total sum of all the feasible assignments values for loans. In order to get the best results using the greedy strategy, algorithm GIA (see figure 4) was implemented and tested with a set of randomly generated problems.

3.1 *APR*

Select the most valuable item per unit weight is a strategy used in the fractional knapsack problem. The fractional knapsack problem is defined as: Given materials of different values per unit weight and maximum amounts, find the most valuable mix of materials which fit in a knapsack of fixed weight. Since we may take pieces (fractions) of materials, a greedy algorithm finds the optimum. Take as much as possible of the material that is most valuable per unit weight. If there is still room, take as much as possible of the next most valuable material. Continue until the knapsack is full [1, 4]. We can map the knapsack problem to our problem, as we have rectangular objects with a value defined in Eqs. 6 and 13, and a weight given by Eqs. 7 and 14.

We give the i_{th} item its weight by the product $interval_i$ times $amount_i$, corresponding to an area, since we have to patch the area under the discrete function curve F with the rectangular objects. So we could make the choice by selecting the objects in decreasing order of $value_i/weight_i$ that is by its APR.

It is therefore not surprising that if we choose the rectangular object with the largest APR is like taking as much as possible of the material that is most valuable per unit weight. In the long run, this strategy yields very good results, as we can fill up the total area under the curve with the best APR available in each step of the algorithm, resulting in a high value of the total sum of its values; according to Eqs. 6 and 19, the APR times the object weight yields its value. A problem arise when there are more than one objects with the same largest possible APR, in this case we use a limited search algorithm.

3.2 *Interval*

Another element used as selection criteria is the length of the *interval* (i.e. duration of the investment), which is very important as we can fill the total area of a given F choosing the objects by the largest possible *interval*; this strategy yields very good results as we can find large APR values and cover a space that allows to fill the remaining available area by large rectangles instead of small ones.

3.3 *Interval* multiplied by powers of *APR*

The selection criterion used in this greedy strategy is to choose the item with the largest value of *interval* multiplied by APR raised to the i^{th} power, where $i = 0, 1, ..., 15$. This way, our recursive algorithm GIA computes a solution to every integer i. The greedy algorithm GIA computes 16 possible solutions to a given problem and takes the solution with the maximum value. This is the best strategy used; its importance is based on trying different assignments, from the widest item to a tallest one, all of them with a high value of APR and not necessarily with the highest one.

3.4 *APR* using a limited search

A problem arise when there are more than one objects with the same largest possible APR, in this case we use a limited search algorithm. First choose one of the objects with

the same largest APR, solve the $LEFT$, $RIGHT$ and TOP subproblems with the greedy strategy choosing the object with the largest value of APR and the longest *interval*, keeping the sum of values of all the objects picked at each step of the algorithm until the total area is full. Then try with the next object and if the total sum per unit weight is greater than the one we have, the object is marked and continue until there is no more objects with the same APR. The next step is to run the algorithm once more, but now the marked object is selected as the basic greedy assignment and continue with the next level at the TOP subproblem. This way we build an implicit tree where the root is the main problem, then the nodes at the first level are the objects with the largest possible APR, once we choose one of those objects, the next level is constructed by the largest possible APR objects at the TOP subproblem. This procedure is done until there is no room for more objects.

3.5 *APR* and longest *Interval*

From the set of items with the largest APR choose the item with the longest *interval*.

3.6 First and last largest *APR*

This greedy strategy choose the first found item with the largest APR from a set of items with the largest APR, then run the algorithm once more and choose the last found item from the same set. Usually, the first formed rectangle with the largest APR value has a point in the left part of the function and the last formed rectangle has a point in the right part of the function. At the end of the procedure we just take the best of both results.

In order to get better results, the greedy algorithm GIA was modified as follows. When a subproblem has a sequence of points either monotonically increasing or monotonically decreasing, the algorithm GIA changes the greedy selection strategy and solves the subproblem with the best of strategies presented in subsections 3.2 and 3.6, test both and takes the one which yields the best results.

4 Results

Computation of the optimal investment assignments set for a given problem is a non-trivial task for managers, specially if the number of points in F is large and the available investment or loan options grow in size.

Since we are proposing an approximation algorithm, its performance is measured through a ratio between the solution value found by the algorithm GIA and the optimal solution value from the brute force procedure; this ratio is represented in percent, showing how close our solution is to the optimal and will be called optimality ratio.

In order to test our procedure performance, we developed an implementation (Java applet available at http://lsc.fie.umich.mx/∼javila/g-applet.html) of the proposed algorithm and greedy strategies mentioned in this paper. As we could not find a similar work to compare our results, we run a search with the brute force approach to get the optimum

from a set of 1000 randomly generated test problems. To date the brute force implementation presented in this paper has solved 416 from all test problems; as problems grow in size its complexity increases and an exponential-time to solve them appears.

For an example of this, the solution of a function with 24 points and 2 peaks using semi-exhaustive search, explores 11,776'527,164 nodes and generates 3,589'656,852 possible solutions. In terms of time, exploring this amount of nodes represents a computational time of 71 days in a PC with a Pentium III processor at 797 MHz. in a LINUX platform and a software programmed in Lisp, using the same hardware, our greedy approach, programmed in Java takes just 1.41 seconds and the optimality ratio is 99.997%. The statistics for the results are shown in table 1, and the performance of heuristic strategies in table 2.

Optimality Ratio Average μ	99.89%
Standard Deviation σ of Optimality Ratio	0.307
$\mu \pm 1\sigma$	94.9%
$\mu \pm 2\sigma$	97.3%
$\mu \pm 3\sigma$	98.5%
Optimal Solutions	40%

Table 1: Statistical Results from Data.

Heuristic as selection criterion	Optimality ratio average (%)	Contribution to solution (%)
Interval multiplied by powers of *APR*	99.84	81.2
APR using a limited search	98.12	15.5
APR and longest *interval*	97.68	1.3
First and last largest *APR*	97.37	2.0
The best of the above	99.89	100.0

Table 2: Performance of Heuristic Strategies.

The efficiency of each heuristic (see table 2), is measured through a ratio between the contribution to solution and the time it takes to produce its results. According to the contribution to solution is the efficiency of the presented heuristics.

We plotted the optimality ratio vs. explored nodes and is shown in figure 6. This plot gives us a clue on the procedure performance. Note that our results are very explicit and the trend does not fall as the number of explored nodes increases. Figure 7 shows a histogram with the optimality percentage frequency in the 416 solved problems by the semi-exhaustive procedure.

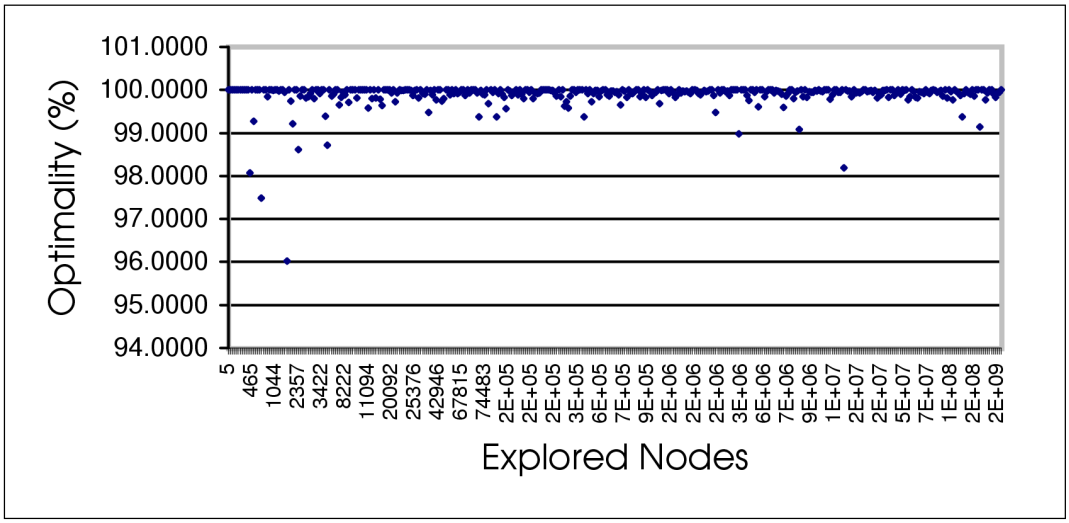

Figure 6: Optimality Ratio Vs. Explored Nodes in Experimental Data.

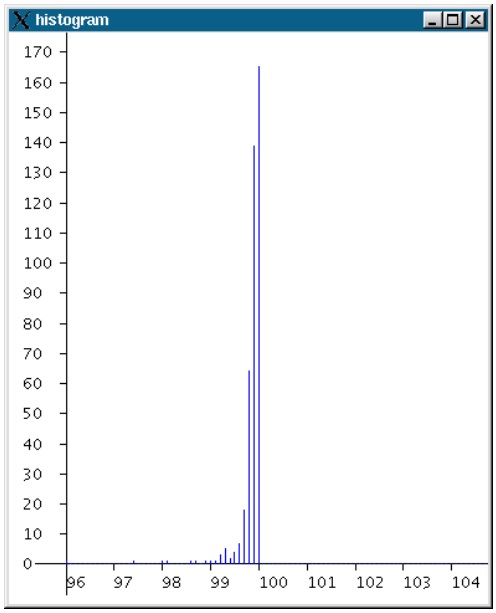

Figure 7: Optimality Ratio Frequency in Experimental Data.

5 Conclusions

In our problem, the number of objects or items of each type is unbounded, there could be as many objects as feasible assignments could fit the available area of a problem or subproblem at each step of the recursion. There is an analogy between our problem and the unbounded knapsack problem (an NP-hard combinatorial optimization problem [4]), in both problems, given items of different values and weights we have to find the most valuable set of items that fit a container with a fixed weight. In the former not just the items weight has to be considered but the shape to fit. Then we could see our problem as a puzzle instead as a knapsack or a bag, where the objects to fill the puzzle are items of different values, weights and shapes.

A brute force approach was used in order to solve the experimental randomly generated problems. This brute force search finds the optimum value trying every possible solution and is exponential in the input length.

The best heuristic used as selection criterion is *Interval* multiplied by powers of APR (see table 2), this heuristic has the best optimality ratio average, the greatest contribution to solution and, as one might expect, is the most efficient, as mentioned in the previous section. Note that for very large-size problems the number of powers must be modified to a greater number (see subsection 3.3).

To evaluate the performance in the search strategy and greedy functions used in the procedure to find the best mix of investment options, we cite some criteria used by [5]:

Completeness: is the strategy guaranteed to find a solution when there is one? The procedure presented in this work did find a solution in all 1000 randomly generated test problems.

Processing time: how long does it take to find a solution? The estimated time our procedure takes to find a solution to all experimental functions is around 5 minutes.

Optimality: does the strategy find the highest-quality solution when there are several different solutions? In a sample of 416 randomly generated test problems of different sizes, our strategy finds an optimality ratio average of 99.89%.

An extension of this paper will be to deal with uncertainty; in this paper we assume a crisp planning horizon, we are aware that this limitation could be improved by using fuzzy mathematical techniques [6].

Acknowledgment

The authors wish to thank Jesús Arellano and Pedro Chávez for their valuable contribution with the Lisp implementation.

References

[1] Brassard G.; Bratley, P. (1997) *Fundamentos de Algoritmia.* Prentice-Hall, Madrid.

[2] Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; Stein, C. (2001) *Introduction to Algorithms.* The MIT Press, Cambridge, Mass.

[3] Flores, J.; González, F.; Flores, B. (2003) "Greedy determination of investment options", preprint, Universidad Michoacana de San Nicolás de Hidalgo, Morelia, México.

[4] Martello, S.; Toth, P. (1990) *Knapsack Problems Algoritms and Computer Implementations.* John Wiley & Sons, Chichester, England.

[5] Russell, S. J.; Norvig, P. (1995) *Artificial Intelligence: A Modern Approach.* Prentice-Hall, New Jersey.

[6] Terceño, A.; de Andrés, J.; Barberà, M.G. (2001) "The use of fuzzy programming for the management of immunised fixed income portfolios, in: *Fuzzy Set Systems in Management and Economy.* World Scientific, Singapur.