Combinatorial Aspects of Yield Management, a Reinforcement Learning Approach

Bruno Martin

Bouygues SA, 1, av. Eugne Freyssinet, 78061 St Quentin en Yvelines Cedex, France
bmartin@bouygues.com

February 5, 2004

Abstract

KEYWORDS: television advertising, multi-knapsack, Monte-Carlo, approximate dynamical programming, revenue management

Yield management is an important issue for television advertising. Anyway, the major part of the research in revenue management focus on the airline or hotel industry. The TV advertising case has some specificities, where the most important is the decomposition of the offer into a lot of small TV breaks (about 10 spots only). The aim of this article is to understand how this changes the optimization policy and propose generic solutions based on simulations and approximate dynamical programming.

1 Introduction

Originally started in the airline industry, Revenue Management is a practice which has found uses in industries and services ranging from car rentals, hotels, cruises, railroads, shipping to the entertainment industry. All activities producing perishable goods or services are concerned.

For TV advertising, revenue management is a major issue: a TV break is a perishable good which has to be sold at the best price. The revenue management problem depends on the regulation and can take different forms in different countries. Let us describe the problem as it occurs in France. Advertising time is limited to 144 minutes per day and 12 minutes per sliding hours. The publicity management opens about 40 TV break each day respecting these constraints. The size of a TV break can reach 360 seconds but its typical size is 240 seconds. Traditionally, there is no price variation during the booking process except 3 weeks before diffusion to be adjusted to the programs expected audiences. Some reductions are proposed to announcers booking late in the remaining space (3 to 1 week before the diffusion). Thus unlike the airline industry, the most contributing clients book first. But the major difference is that the size of a TV break (typically 240 seconds) is relatively small compared to a TV spot length (from 5 to 45 seconds with an average of 20 seconds).

Developed for the airline industry, Yield Management (Belobaba and Williamson 1988) is today the major Revenue Management approach. Good solutions to deal with the single-leg problem are well known (McGuill and Ryzin 1999), the idea is to compute the Expected Margin (Seat) Revenue of a client depending on its fare, its cancellation probability and the probability to get a more valuable client instead. An active research is done on this topic to find good heuristics in multi-leg problems (e.g. Alstrup *et al.* (1986), Berstimas and Popescu (2003), Gosavi *et al.* (2002) and Karaesmen and Ryzin (2004)).

The model presented in Benoist *et al.* (2001) is well adapted to study TV advertising specificities. It is an hotel reservation problem facing only group reservations. The similarity with TV advertising comes from the low capacity of the hotels and the various sizes of the groups. They have shown that classical yield management approach is outperformed by other solutions in this particular combinatorial case. The problem is such that the hotels capacity are of 5 small groups and it is necessary to deal with lost capacity: if the number of free rooms is lower than the size of the groups, the rooms are lost. We have to deal with a knapsack problem. Actually, the yield management approach assume that the reservation process is fluid: an elementary transaction is small with respect to the available capacity. Of course, this assumption is perfectly legitimized in most real cases including the airline industry: a few seats among a few hundred seats.

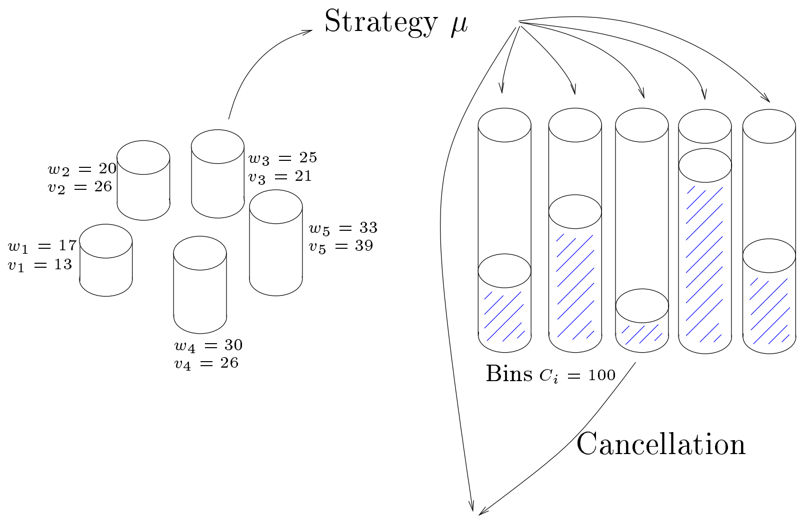

Figure 1: At each time step some items arrive and some cancel. The strategy decides to accept the incoming items on one of the bins or to reject them. At the end, the score is the sum of the values of the items present in the bins minus some penalties in case of over-book.

The purpose of this article is to explore Revenue Management problems when the fluidity assumption cannot be made. In this case, good methods have to deal with the combinatorial aspect of the problem. We first describe the formal model and previous results (Benoist *et al.*, 2001). We then present two successive improvements. First using low discrepancy sequences with Monte Carlo and then using a new method based on reinforcement learning. This last method gives the best results and improve by 1.8 % the previous benchmark. It appears that in practice, Monte-Carlo and Quasi-Monte-Carlo cover most cases with result at 1 % of the reinforcement learning strategy. Both solutions can be easily adapted to a wide range of stochastic multi-knapsack problems.

2 Model description

To focus on the combinatorial aspect of the TV advertising problem, we will study the model proposed by (Benoist *et al.*, 2001). It is a generalization of the stochastic (single) knapsack problem which is well studied in (Kleywegt and Papastavrou 1998) for single size items and (van Slyke and Young 2000) for the multi-size and multidimensional case. Here we are interested in the multi-size and multi-knapsack problem. Due to the multi-knapsack aspect, dynamic programming is not possible. This model is generic, small enough to allow simulations and thus benchmarking, and highly combinatorial so that the problem cannot be solved optimally even for small numbers of bins.

The problem is modeled as an on-line optimization process with discrete time and deadline. Assume we dispose of N bins of capacity C_i that we can fill with items. There are K different types of items (characterized by a size w_k and a value v_k). At the end of the booking process, the revenue is the sum of the values of the items still presents in the bins minus a penalty if one or many bins are over-booked.

The booking process is the following: at each time step, the operator receives a sequence of booking and/or cancellation requests. There is nothing to do with cancellation except to update the bin availability. At each booking request, he has to answer immediately if this request is accepted or rejected and in case of acceptance, in which bin it is accepted. What we are looking for is a strategy $\mu(t, H, S, i)$ which is function answering *reject* or a bin number from the current period t, the past arrivals and departures H, the current content of the bins S and the type of the incoming item i (see Figure 1). We know both the probability law of arrival and cancellation from now until the end and the usual distribution of these arrivals and cancellations among the remaining periods. The expected revenue of a strategy can be approximated throw simulations and the objective is to find the strategy with the best expected revenue.

As the problem is on-line, we consider only the strategies answering in less than one second.

To resume, the model is an on-line multi-choice knapsack with deadlines. At each period, we have

Problem	Comment	BF	FS	EV	CA	YM	best gain	CB
Master		454	491	487	489	465	8.2 %	542
Pb1	smaller penalty	454	500	494	498	485	10 %	542
Pb2	various q_k	469	510	504	509	482	8.7 %	556
Pb3	bigger items	431	459	449	458	380	6.5 %	508
Pb4	single knapsack	82	86	83	87	60	5.8 %	101
Pb6	value = $size^2$ (convex)	11782	13175	12951	13232	12630	12 %	14439
Pb7	value =$10\sqrt{size}$ (concave)	910	948	938	948	901	4.2 %	-
Pb8	huge arrival volume : 300%	458	533	538	513	511	17 %	587
Pb9	small arrival volume: 120%	421	425	420	426	393	1.2 %	-
Pb10	no departures	469	542	533	524	514	15 %	-
Pb11	Small items more frequent	442	488	483	487	465	10 %	-
Pb12	Big items more frequent	464	504	504	499	472	8.5 %	-
Pb17	Large standard deviation: 35%	452	486	479	-	464	7.6 %	-
Pb18	Small standard deviation: 12%	453	495	486	481	458	9.2 %	-
Pb19	No deviation and no departure (only the order change)	468	542	536	548	481	17 %	-
CPU	Number of decisions per second on the Master problem	200 000	3	100	1	2 000		

Figure 2: Comparison between 6 strategies on the Master problem and some variations. The best gain is the gain of the best method over the BF (Best Fit) greedy algorithm.

a sequence of arrivals and departures and for each arrival, we have to choose to refuse it or to decide in which bin we put it.

From our experience, discrete time step and prevision hypothesis are rather realistic in an operational context. Usually, forecasts for future incoming and cancellations are made at some key dates in the reservation process. Each forecast improves the previous one by taking into account the number of reservations since the last period. Usually, few received requests suggest a bad conjuncture and few requests to come.

What is less realistic in this model is that the client has no preference among the bins. In TV advertising, announcers ask one specific TV break (e.g. the Tuesday at 20h40 just before the film). Anyway, a prime break is often full and the announcer is likely to accept another TV break of similar price and public (the same day at 20h30 or another day at the same time). Nevertheless, the client can refuse the counter-proposition or cancel it later to better fit its initial choice. To deal with preferences, the probability of cancellation of an item should depend on its bin.

3 Previous benchmark

Let us recall shortly the main results of (Benoist *et al.*, 2001).

The test protocol in (Benoist *et al.* is centered on a Master problem. The proposed strategies were tuned on this problem and then also tested on many variation around this problem to see how robust they are. In this article, we will focus on the Master problem defined below. Then we recall the strategies and the result of the benchmark.

3.1 Master problem

For the benchmark, the master problem is detailed below:

- 5 bins of capacity 100, 10 periods, 5 types of items;
- sizes of items are respectively (17, 20, 25, 30, 33);
- values are respectively (13, 26, 21, 26, 39);
- Overload penalty is 10 (a bin filled at $C_i + n$ has a penalty of $10 \times n$);
- The arrival amount is a binomial law of mean 8 and they are uniformly dispatched over the 10 periods;
- The leaving probability is a memory-less law with initial leaving probability of 50 % (at each period, the probability for an item to leave is 6,7 %);

3.2 Proposed strategies

The Figure 2 resumes the results of the six strategies shortly explained below and detailed in (Benoist *et al.*, 2001).

BF stand for Best Fit, it is a greedy strategy which sort the bins by free available space (in increasing order) and accept the incoming item in the first bin into which it fit without over-book. FS (Forward Sampling) is a Monte-Carlo strategy using BF as slave to evaluate the expected revenue of all possible future decisions. With the same principle, EV (Expected Value) and CA (Combinatorial Analysis) replace the sampling by deterministic hand-made explorations. YM (Yield Management approach) implement the strategy that would be optimal if the problem were completely fluid (McGuill and Ryzin 1999). CB (Crystal Ball) computes an upper bound for the stochastic problem by solving the far-seeing problem (we assume that we know all future incoming and cancellations) using a linear programming approach. It provides an upper bound.

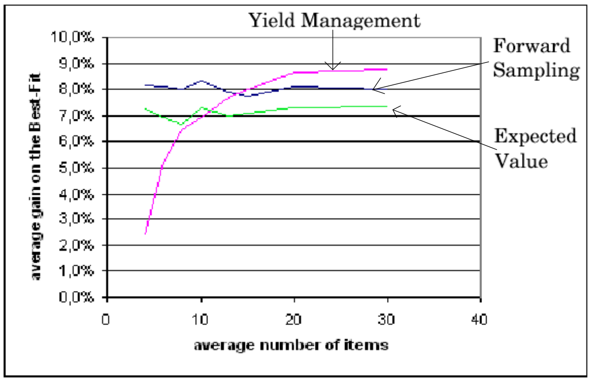

Figure 3: Comparison between Yield Management and combinatorial methods for different sizes of the bins

A very interesting result of (Benoist *et al.*, 2001) is to show the transition between problems suitable for a discrete approach and problems fluid enough to ignore the combinatorial aspect of the problem. Figure 3 show that Forward Sampling method is better than Yield Management method when there are less than 15 item per bin in average, but above 15 items per bin, the Yield Management method is better.

4 Structural properties

Let us recall some well known theoretical properties on the problem that we will use next to improve the optimization strategy.

As defined above, a strategy $\mu(t, H, S, i)$ is a function answering *reject* or a bin number from the current period t, the past arrivals and departures H, the current content of the bins S and the type of the incoming item i. Each strategy has a performance which is the expected revenue if this strategy is used. An optimal strategy is a strategy which maximizes the expected revenue. It is well known that optimal strategy can be expressed as fixed point (Bertsekas 2001):

Theorem 4.1 *Let us choose a strategy μ_0. Define μ_{n+1} the strategy which takes optimal choices if μ_n is to be used in the future. By iterating this process, we asymptotically obtain an optimal strategy.*

The idea of the proof is to introduce the revenue function R_μ associated to a strategy μ, it is the function which gives for all possible configuration (the currents period, the past arrivals and departures, the current content of the bins) the expected revenue if we apply μ. We then use the nice properties of the partial order on the set of expected revenue functions ($R_\mu \leq R_\nu \iff \forall x R_\mu(x) \leq R_\nu(x)$). If R_μ and R_ν are two expected revenue functions, then $max(R_\mu, R_\nu) : x \longmapsto max(R_\mu(x), R_\nu(x))$ is an expected revenue function too (for all decision to take, we choose the best of μ and ν decisions).

5 Quasi-Monte-Carlo

The best method benchmarked in (Benoist *et al.*, 2001) is based on a Monte-Carlo method. This solution has the big advantage to be completely generic and quite easy to implement. It is thus a very good candidate in most revenue management problems with combinatorial aspects. The aim of this section is to study a classical improvement of this method: the Quasi-Monte-Carlo method. The only difference with Monte-Carlo is to use determinist low discrepancy sequences rather than (pseudo)-random numbers to generate the sample of future scenarios. In the next section we will propose a better strategy based on reinforcement learning.

5.1 Low discrepancy sequences

In order to be self-content, let us shortly recall here the low discrepancy ideas and the mostly used sequences (Drmota and Tichy 1997). To numerically integrate a real function f on a give interval $[0, 1]$, we can make the average of $f(x_n)$ if (x_n) is a randomly distributed sequence over $[0, 1]$, but a better method is to take the average of $f((i - 1/2)/n)$ for i in $\{1, ..., n\}$. The second method converges more quickly because the points are regularly distributed on the interval. Low discrepancy sequences are generalization in higher dimension of this principle. Discrepancy measures how badly distributed a sequence of point is, the goal is thus to find low discrepancy sequences. Most of them are defined from Corput sequence (Kuipers, Niederreiter, and Shiue 1975). Let us decompose n in base b: $n = \Sigma d_i b^i$, we define $C_{b,n} = \Sigma d_i b^{-i-1}$. Now, in dimension k, let us choose b_1, b_2, , b_k, k coprime positive integers. We define Halton and Hammerslay sequences as follow (Cavior 1977a), (Cavior 1977b):

$$Halton \qquad n \longrightarrow (C_{b_1,n}, C_{b_2,n}, C_{b_k,n})$$

$$Hammerslay \qquad n \longrightarrow \left(C_{b_1,n}, C_{b_2,n}, C_{b_{k-1},n}, \frac{n}{N}\right)$$

where N is the number of points we need. We can see in dimension 2 the better repartition of Halton sequence over randomly chosen points on Figure 4.

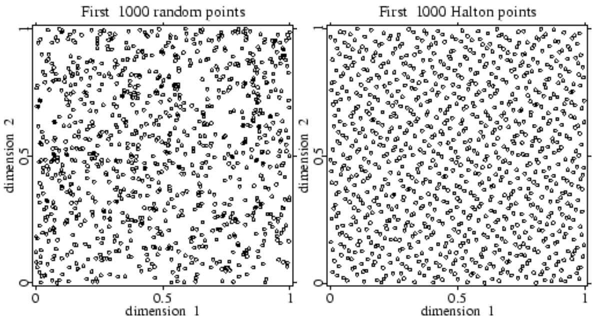

Figure 4: Comparison between random points and Halton points in dimension 2

We can prove that while random numerical integration converge as $O(1/\sqrt{n})$, Halton sequence allow $O((logn)^k/n)$ convergence and Hammerslay $O((logn)^{k-1}/n)$. These sequences are asymptotically better, but are practically unusable when the dimension increases (the minimal number of points to have a good repartition in all dimensions increase exponentially with k).

5.2 Quasi-Monte-Carlo benchmark

It is not feasible to replace all random number accesses of the online multi-choice knapsack by one low discrepancy sequence of high dimension. Anyway, the major uncertainty parameter is the number of items to arrive of each kind. We thus modified Monte-Carlo method to use a low discrepancy sequence in dimension $k = 5$ to generate the number of items to arrive for each of the five types. The other random choices (when they will arrive, and the cancellations) are still made through a random generator.

The following array (Figure 5) summarizes the result obtained on the master problem. The decreasing performance of Hammerslay sequence is a measure error: due to the computation time, the presented results of the rightmost column are the average result on only 1000 scenarios.

Number of iterations	100	200	1000
Monte-Carlo	490		491
Halton	492	494	494
Hammerslay	494	492	493

Figure 5: Benchmark of Quasi-Monte-Carlo method.

We observe that the number of samplings (less than 1000 for speed reasons) is too low to obtain the very impressive speedup observed in financial mathematics (Paskov 1994). Anyway, despite the low confidence of the results, we see that Quasi-Monte-Carlo obtain better results with less iterations. It appears as a low cost way to improve and/or speedup a simulation based strategy.

6 Reinforcement learning method

We now use the theoretical results of Section 4 to find a strategy better than previously defined Forward Sampling strategy based on Monte-Carlo and Quasi-Monte-Carlo methods. The error made by Forward Sampling is to assume that the Best Fit strategy will be used in the future. Actually we will in the future use the Monte-Carlo strategy with Best Fit as slave (which is a better strategy). For computation time consideration, it is impossible to use a Monte-Carlo method on the Forward Sampling strategy (each decision would take almost 6 hours). Thus the basic idea is to store the Monte-Carlo strategy and use it as a new slave. We can then iterate the process and theoretically, this converges to the fix point policy which is optimal (see section Structural Properties). Unfortunately, the information needed to store a policy is too big and we have to approximate this policy. We obtain a classical reinforcement learning policy described in (Bertsekas 2001), its principle is explained on Figure 6.

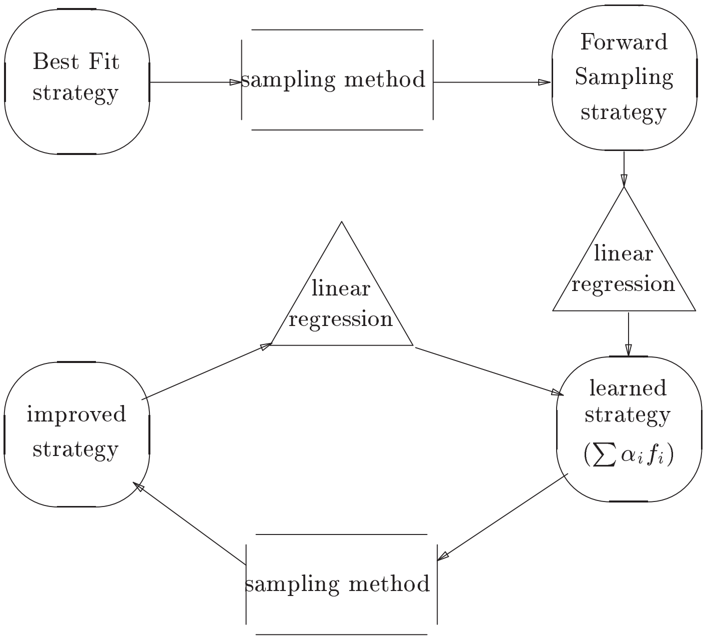

Figure 6: The Approximate Policy Iteration method. From a fast strategy (Best Fit), we can take a better decision by simulating the consequences of all possible choices to take the best (Forward Sampling). This method is slow but can be approximated by a fast method (learned strategy). While the new fast method is better than the previous one, we can iterate the process.

6.1 Choice of the policy approximation

As we know, we can associate to a policy μ an expectation value function R_μ which says for each configuration the expected value if the policy is to be used in the future. It appears obvious that from an expectation value function we can compute the initial policy by evaluated each choice consequences and choosing the one which lead to the configuration of maximal expected value. This expectation value function is usually easier to approximate.

The problem is thus to find a function of low Kolmogorov complexity which approximates the function from a configuration (the currents period, the past arrivals and departures, the current content of the bins) to $\mathbb{R}$ obtained by a Monte Carlo simulation. The simplest method is to approximate such a function by a linear function $\sum \alpha_i f_i$ where f_i are functions extracting significant parameters from the configuration and α_i are real parameters which will be adjusted optimally to the function to approximate thanks to a linear regression. Of course, the problem now is to find good functions f_i.

To understand the role of f_i functions, one could imagine we want to make a very basic chess software. To choose the next computer move, we will make a tree exploration at some depth and then choose the move which maximize the final position whatever the opposite player does. To implement this strategy, we need a function which evaluates any chess positions. A kind of "expected winning probability" starting from this position. There are far too many chess positions to store a value for all of them and a very basic approximation would be to give some points to each piece and make the difference of points between the two players. In this case, f_1 could be the difference between our number of pawns and the opponent number, f_2 for rooks, f_3 for queens... We could then imagine that the values of the pieces could be obtained by a linear regression from statistics on a huge amount of played games.

Thus, our problem is to choose the f_i functions to approximate how good is a configuration. Let us first define the functions we took and then explain how it works and the reasoning that led us to these functions.

6.1.1 Choice of f_i functions

We defined two groups of functions, in the first group, we have K functions (K is the number of item types) and in the second group, we have $C + W + 1$ functions (C is the max of the sizes of the bins and W the max of the sizes of the items) defined as follow:

f_i	α_i
Item of size 17 and value 13	6.6
Item of size 20 and value 26	13.2
Item of size 25 and value 21	10.7
Item of size 30 and value 26	13.7
Item of size 33 and value 39	19.9
Empty bin	93.7
Bin with 83 room left (100 - 17)	86.1
Bin with 80 room left (100 - 20)	81.3
Bin with 75 room left (100 - 25)	80.9
Bin with 70 room left (100 - 30)	77.8
Bin with 37 room left (100 - 33)	75.0
Bin with 63 room left (100 - 17 - 20)	72.4
Bin with 60 room left (100 - 20 - 20)	68.0
...	

For instance if all bins are empty the expected value is:
$5 \times 93.7 = 468.5$

If one item of size 20 is booked, the expected value is:
$13.2 + 4 \times 93.7 + 81.3 = 469.3$

If an item of size 17 arrive we can put it
– in the same bin as the item of size 20, we get
$6.6 + 13.2 + 4 \times 93.7 + 72.4 = 467$
– or in an other bin with expected value of:
$6.6 + 13.2 + 3 \times 93.7 + 86.1 + 81.3 = 468.3$
Thus we should refuse the item because the expected value if we accept it (467 or 468.3) is lower than the initial expected value (469.3).

Figure 7: An example of using the α_i and f_i to make decisions. The expected values are those of the best learned strategy at first time step[1].

- for $i \in \{1, 2, ..K\}$, f_i is the number of items of type i currently present in the bins;
- f_{K+j} for $j \in \{-W, -W+1, ..., -1, 0, 1, .., C\}$ is the number of bin in which there are exactly j free space (negative j is for over-booking possibility). Note that the sum of these functions is N (the number of bins) and thus at most N functions have a non null value.

These functions do not depend on the current period because the idea is to do one linear regression by period (with the same set of functions). Thus with the same f_i functions, we compute at each time step a family $\alpha_i(t)$. This is a classical way to decompose the approximation problem (Bertsekas 2001).

Furthermore, these functions depend only on the current state of the bin and ignore the historic data on the past arrivals. The influence of past arrivals on the current situation evaluation is to change the future arrival probability. Anyway, this information is partially redundant with the current filling of the bins and we thus made the choice to ignore it.

Our intuition is that the expected value of a configuration can be decomposed in two parts. On the one hand the expected value of the already accepted items (some will be canceled, but the remaining ones will make revenue). On the other hand the expected filling of the free space in the bins. Obviously, at early periods, there are few accepted items and the value of a configuration is mainly the hope of future booking, while near the deadline, the expected value essentially depends on the items already booked. See Figure 7 for an illustration[1].

The first group of functions is thus meant to approximate the value of the accepted items. The idea is that this value depends on the values of the accepted items reduced by their respective leaving probability, whatever the bin they are affected to. The K functions allow the linear regression to adjust the α_i coefficient to give a different value to each kind of item, depending on the period (in time, the value of an item increase since it has a lower probability to leave).

The second group of functions has to estimate the expected value of future bookings. This depends on the period (the total amount of future incoming items) and the free space we have to accept them (if all the bins are full, we cannot hope many incoming booking). Furthermore, we have to deal with the discrete aspect: it is better to have 25 free rooms in one bin than 15 and 10 in two different bins. Note that the over-book case is treated because negative expected values corresponding to the expected probability can be affected by the linear regression for negative values of free space. It is not obvious that the expected value of two bins with n free space is twice the expected value of only one bin with n free space. The intuition is that some diversity among the bins can be useful. This hypothesis is easy to check (by multiplying the number of f_i functions by the number of bins N, the new functions have values in $0, 1$ and the meaning "there is i bins with j free space"). The experimentation confirms the hypothesis with a very good precision.

[1] Due to conditioning reasons, the α_i obtained by regression are less intuitive (154105.9, 161188.3, ...). For the reader convenience, we propose smaller α_i values. They are computed to give the same results on the examples.

Optimization method	Result on the Master problem	Gain over BF	CPU (number of decisions per second)
BF (Best Fit – best greedy algorithm)	454	-	200 000
FS (Forward Sampling – best strategy)	491	8.1 %	3
FS low discrepancy	**494**	8.8 %	20
EV (Expected Value – fastest strategy)	487	7.2 %	100
RL (Reinforcement Learning) learned strategy	**489**	7.7 %	100 000
RL Monte-Carlo with learned strategy as slave	**500**	10.1 %	5
CB (Crystal Ball)	*542*	*19.3 %*	

Figure 8: Comparison between the proposed methods (Forward Sampling with low discrepancy, Reinforcement Learning methods) and the previous results.

6.1.2 Computation time considerations

The reinforcement learning method needs the learned strategy to be very fast to compute. The chosen approximation method (linear regression) grants that the result is obtained by a scalar product and it is thus computed very quickly. Nevertheless, a scalar product of size about 100 is too slow when you need 6 scalar products to compute the decision (we need to compare the expected revenue after each possible choice). A first optimization is to consider that most of the second group f_i are null, but it is still too slow. The result we present are obtained by computing only the differences between the current state evaluation and each of the 6 possible choices. Due to the f_i structure, it is a scalar product of size 3 (we add one item of the received type and make the difference between the current and future free space in the chosen bin).

7 Results

We used the reinforcement learning principle, starting from the FS (Forward Sampling) method, we learned a strategy (using a linear regression with the previous f_i functions). We then launched a new Monte Carlo method (with 100 to 1000 iterations using low discrepancy sequence) with the learned strategy as slave, learned this new strategy and so on. Typically 5 steps are enough to get a strategy that cannot be improved with the method. It appears that some luck is needed to find a good learned strategy (as it is a pseudo-random process, it can be useful to try different iteration number and keep the best result).

The results are presented on Figure 8. On the master problem benchmarked in (Benoist *et al.*, 2001), we obtained 489 with the best learned method and 500 with a Monte Carlo method using the previous one as slave. Thus the learned strategy is 7,7 % better than the BF. It is also better than EV and 1000 time faster. Furthermore, the Monte Carlo method allow to earn 1.8 % over the previous best result (500 for RL with Monte Carlo vs 491 for FS) with comparable speed (thanks to low discrepancy sequence, we can reduce the number of iterations).

We tested many variant of f_i families and the chosen one is both the simplest and the most effective. Based on simple ideas (give an expected value to present items and free room) it can be adapted to numerous revenue management problems. At last, it is always very fast to compute because most terms of the scalar product are null by definition, which appears to be very important to make the Monte-Carlo method on the learned strategy fast enough.

8 Conclusion

The lack of fluidity on the booking process changes the revenue management approach. Yield management classical solutions are outperformed by specific techniques dealing with the combinatorial problem.

From the study of the model introduced in (Benoist *et al.*, 2001), two main solutions are of interest: (Quasi)-Monte-Carlo and reinforcement learning. The first is very easy to implement. Once found a fast greedy heuristics, it is used to evaluate by simulation the expected revenue of each choice. As we show, the introduction of low discrepancy sequences is easy too and can significantly improve the quality of the solution and/or the computation time. Reinforcement learning approach improves the Monte-Carlo solution. The difficulty is to find a good strategy approximation method. Anyway, the linear regression proposed in the article appears to be fast, simple and generic. Nevertheless, the reinforcement learning approach is less flexible since learning has to be done for all possible arrival law.

Little research has been done on the combinatorial aspects of revenue management. We show that it has a real impact on the revenue to adopt specific approach. The mathematical and computational method are well known from the stochastic optimization, dynamic programming and control research and the adaptation of these method to the specificity of revenue management problem appears as a promising research area.

References

Alstrup, J., Boas, S., Madsen, O. B. G., and Vidal, R. V. V., 1986, Booking policy for flights with two types of passengers, *European Journal of Operational Research* 27, 274–288.

Belobaba, P. P., and Williamson, E. L., 1988, Optimisation techiques for seat inventory control, in *Proceedings of AGIFORS*.

Benoist, T., Bourreau, E., Caseau, Y., and Rottembourg, B., 2001, Towards stochastic constraint programming: A study of on-line multi-choice knapsack with deadlines, in *Proceedings CP'01* , vol. 2239 pp. 61–76. Lecture Notes in Computer Science Springer Verlag.

Bertsekas, D. P., 2001, *Dynamic Programming and Optimal Control Volume II (Second Edition)* (Athena Scientific).

Bertsimas, D., and Popescu, I., 2003, Revenue management in a dynamic network environment, *Transportation Science* 37, 257–277.

Cavior, St. R., 1977a, Uniform distribution for prescribed moduli, *Fibonacci Q.* 15, 209–210.

———, 1977b, Uniform distribution (mod m) of recurrent sequences, *Fibonacci Q.* 15, 265–267.

Drmota, M., and Tichy, R. F., 1997, *Sequences, discrepancies and applications* . , vol. 165 (Lecture Notes in Mathematics).

Gosavi, A., Bandla, N., and Das, T. K., 2002, A reinforcement learning approach to airline seat allocation for multiple fare classes with overbooking, *IIE Transactions* 34, 729–742.

Karaesmen, I., and van Ryzin, G., 2004, to appear, Overbooking with substituable inventory classes, *Operations Research*.

Kleywegt, A. J., and Papastavrou, J. D., 1998, The dynamic and stochastic knapsack problem, *Operations Research* 46, 17–35.

Kuipers, L., Niederreiter, H., and Shiue, J., 1975, Uniform distribution of sequences in the ring of Gaussian integers, *Bull. Inst. Math., Acad. Sin.* 3, 311–325.

McGuill, J.I., and Ryzin, G.J. Van, 1999, Revenue management: research overview and prospects, *Transportation science* 33.

Paskov, S., 1994, Computing high dimensional integrals with applications to finance, in http://citeseer.nj.nec.com/paskov94computing.html.

van Slyke, R., and Young, Y., 2000, Finite horizon stochastic knapsacks with applications to yield management, *Operations Research* 48, 155–172.