Towards Stochastic Constraint Programming: A Study of On-Line Multi-Choice Knapsack with Deadlines

Thierry Benoist, Eric Bourreau, Yves Caseau and Benoît Rottembourg

Bouygues e-lab, 1 av. Eugène Freyssinet,
78061 St Quentin en Yvelines Cedex, France
{tbenoist,ebourreau,ycs,brottembourg}@bouygues.com

Abstract. Constraint Programming (CP) is a very general programming paradigm that proved its efficiency on solving complex industrial problems. Most real-life problems are stochastic in nature, which is usually taken into account through different compromises, such as applying a deterministic algorithm to the average values of the input, or performing multiple runs of simulation. Our goal in this paper is to analyze different techniques taken either from practical CP applications or from stochastic optimization approaches. We propose a benchmark issued from our industrial experience, which may be described as an On-Line Multi-Choice Knapsack with Deadlines. This benchmark is used to test a framework with four different dynamic strategies that utilize a different combination of the stochastic and combinatorial aspects of the problem. To evaluate the expected future state of the reservations at the time horizon, we either use simulation, average values, systematic study of the most probable scenarios, or yield management techniques.

1. Introduction

One of Constraint Programming (CP) major claims to success has been its application to solving complex industrial application problems. The strength of CP is its ability to represent complex domain-dependent constraints, which yield interesting modeling abilities, that have been completed, over the previous years, with resolution techniques such as meta-heuristics [CLS99]. Interestingly, most industrial combinatorial problems are stochastic in nature, due to the uncertainty that is characteristic of real world situations. Our own experience with industrial problems include construction planning, call center scheduling, equipment inventory management or TV advertisement booking, to name a few. In all these situations, the resolution of a static problem is only a compromise, since the data that is given to the algorithm only reflects the situation that is expected in the future. As a consequence, the algorithm needs to be run often and incrementally, to adjust to real-time events that modify the validity of the solution. When the future is widely unpredictable, this may be the wisest thing to do, but most often, probabilistic information is available (mean, standard deviation and distribution characterization), that should be used to derive more robust solutions.

When confronted with this challenge, practitioners of constraint programming have often developed and used ad hoc techniques. There is a large lore of practical advise

available in the community, but very little of it has been formalized. Three strategies may be observed:

1. The simplest is to use a static combinatorial algorithm using the expected values (means) as the input.
2. Another simple approach is to use simulation to compare possible decisions, making multiple runs of the algorithm using different sets of input data that are generated following the probability information that is available. This is easy to implement but places a heavy burden on the run time.
3. Hybrid approaches may be developed, that try to introduce the stochastic nature into the general design of the algorithm. For instance, a strategy that was reported to the authors in the field of industrial process control is to couple a CP solver that solves a resource-constrained scheduling problem with a stochastic generator that produces "ghost" orders in the future and with a matching algorithm that links upcoming orders with ghosts when appropriate. Thus the scheduler runs continuously (and incrementally) on a flow of tasks that contains both real and predicted orders.

The nearby field of stochastic optimization has produced over the last 30 years a wealth of impressive results, some of which being very relevant to this issue ([BL97] for a survey). These results both cover the stochastic behavior of classical combinatorial algorithm or heuristics (such as the WSPT rule from W. Smith) or the definition of new, stochastic, algorithms [HSS+96,ABC+99]. However, these results rely both on the relative simplicity of the static algorithm that is analyzed stochastically, and many independence hypotheses about the input data that are required to make the analysis work.

The combination of NP-hard problems and stochastic input data has been studied under the field of *online algorithms* [FW97]. Competitive analysis has provided some upper bounds for the *competitive ratio*, which is the ratio between the worst-case behavior of the online algorithm and the optimal solution given a complete knowledge of the input. However, there are few practical results and fewer practical good heuristics available for these problems. The consequence is that *simulation* is required for answering most of the characterization questions that we may have, as soon as the problem becomes too difficult. For instance, finding the probability that the makespan of a simple scheduling problem with precedence (that would be solved statically with a PERT) is higher than a given value is achieved with a Monte-Carlo simulation (more precisely, with a pseudo-MC simulation [AW96]). The scheduling problems that we address in our industrial applications are NP-hard (contrary to the PERT problem) and the stochastic characterization of the complex branch-and-bound algorithms that are used to solve them is an open issue.

Our goal with this paper is to propose a first contribution that creates a bridge between the two communities. On the one hand, we want to address combinatorial problems with complex constraints while taking the stochastic nature into account. On the other hand, we want to follow a scientific approach, using public benchmarks, and a survey of different techniques that either come from the field of stochastic programming or from the experience of CP practitioners. Thus our goal is threefold:

1. Compare the relative strengths of different techniques that we have used in the past on different problems, as well as methods that we have been referred to by researchers from the stochastic programming community, including "Yield Manage-

ment". YM has been popularized by the airline industry and is finding its way into many industrial domains.
2. Propose directions that may lead to better algorithms, either through the stochastic characterization of complex CO techniques or through the introduction of "combinatorial insights" into stochastic methods.
3. Publish a benchmark problem that is representative of our industrial problems and holds a strong combinatorial component, yet that is simple enough so that many other methods can be applied and compared.

The experience that we have proposed as a benchmark comes from an online reservation system. It has been simplified and can be described as a multi-choice knapsack with deadlines. It represents the reservation problem faced by a tour operator that operates a set of sites and receives reservation requests, as well as cancellations, for groups of tourists. The goal is to maximize the final occupancy of the hotels (at a given deadline), given a penalty that is given for overbooking. The algorithm that needs to be built receives the stream of reservation/cancellation events and is required to accept or decline each reservation.

The paper is organized as follows. The next section gives a detailed presentation of our benchmark problem and an associated best-fit dynamic algorithm that will be used as a reference. The algorithms presented in this paper are dynamic algorithms that computes, each time they are called, an expected valuation of the current state with and without the incoming reservation. The difference between the various approaches is the technique that is used to make this valuation. Section 3 presents a set of approaches that are based on simulation, using a reasonably simple strategy to fill the hotels (bins). Section 4 presents on the contrary an approach that uses a more sophisticated filling algorithm but that is only applied to mean values. Section 5 presents a technique that is based on the characterization of the most representative scenarios of demand/cancellation and then solves heuristically the problem for each scenario. Section 6 presents a Yield Management approach, which is based on the characterization of the expected marginal revenue over the duration of the scenario. The last section gives a preliminary comparison of the results of these different approaches.

2. Problem description

2.1. Mathematical description

Let b_1, b_2, … b_N, be N bins of respective capacities C_1, C_2,…C_N and consider K types of items of sizes w_1, w_2, .. w_K, and values v_1, v_2, .. v_K. The numbers of items of each type are positive integer random variables In_k. The arrivals of these items are distributed into T periods indexed from 0 to T-1, according to a common repartition function $f(t)$. During each period, previously arrived items have a probability q_k to leave.

A state of the packing is an N x K matrix P of positive integers, such that $P[n,k]$ is the number of items of type k present in bin b_n. A strategy is a function $s(P,k,t)$ returning an integer in [0,N], denoting in what bin an item of type k arriving at date t should be put (0 standing for refusal).

All bins are initially empty. At each date t ∈ [0,T-1], a list of events is presented one after another, in random order. These events can be :

- The departure of an item of type *k* (that was present at date t) from bin *n*: then P[n,k] is decreased by one
- The arrival of an item of type *k*: then if *s(P,k,t)* is non null, P[*s(P,k,t)*,k] is increased by one.

The objective of the strategy is to maximize the expected value of the following function at the end of the process:

$$F=\sum_{n=1}^{N}\left(\sum_{k=1}^{K} v_k P_{n,k} - \alpha \times \max\left(0, \sum_{k=1}^{K} w_k P_{n,k} - C\right)\right) \quad (\mathbf{1})$$

where α is a penalty coefficient.

2.2. Corresponding situation

This problem could be that of a travel agency, filling holiday centers with school groups, in presence of stochastic demand and cancellation. In this case, bins and items would be respectively holiday centers and stays of school groups. We consider the optimization of the filling during a specific week (say Christmas week). Stays are booked by school academies, each of them characterized by the size of the groups it organizes (deterministic) and the price it pays for each group (not necessarily related to the group size). Until Christmas, these academies can book stays for this festive week, but they *cannot* specify in which holiday center they would like to go. On the other hand, when the agency accepts to book a stay, it informs the group of its destination, and cannot modify it later. School academies can also cancel previously booked stays (with no penalty). The goal of the agency is to maximize its final profit under the following overload constraint: for each holiday center, each supernumerary person must be accommodated in a nearby hotel at fix cost α.

2.3. Scenarios

In all considered scenarios, bins have identical capacities (normalized to 100) the repartition function *f(t)* is uniform and arrivals follow simple integer distributions (binomial). Experiments were conducted on more than 20 scenarios, based on a "master" scenario detailed below:

- 5 bins, 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 α=10 (see equation (**1**))
- For all k: q_k=0.066967 (→initial leaving probability = 0.5) and In_k= $\mathcal{B}_{12,\frac{8}{12}}$ (→average=8)

2.4. Naïve strategies and Far-seeing strategies

Some strategies do not take into account the stochastic aspects of the problem. For instance the First-Fit (also named First-Come First-Serve) consists in putting arriving

requests in the first possible bin (if any). Otherwise it is refused (no overbooking). The Best-Fit strategy is a refinement of the First-Fit: the arriving item is put in the possible bin with the smallest current remaining size. This Best-Fit strategy (BF) will be used in the following to normalize results on different scenarios.

A far-seeing or clairvoyant strategy is a strategy that knows exactly what will happen in the future, and deals therefore with a deterministic problem. Hence it is *not* a feasible strategy but only a hypothetic one that can be useful to compute upper bounds. In our case a far-seeing strategy knows exactly what requests will arrive, at which dates, and when they will be cancelled. Then it can extract the number of requests of each type that will not be cancelled before the final date T. Hence the associated deterministic problem is a *multi-choice knapsack*. This deterministic problem can be optimally solved using integer programming (less than 5 mn with XPRESS-MP).

3. Forward sampling (FS)

As explained in the introduction, our dynamic algorithms are based on state evaluation. When a request arrives, each possible decision leads to a different state. After evaluating all states with a certain method, the decision leading to the best state is chosen.

A simple idea to evaluate a state is to generate scenarios from the current date to the end (date T), or more precisely from the beginning of the next period to the end of the final one (period T-1). Hence the evaluation of a state will be an aggregation (usually the average) of the evaluations of "many" generated scenarios (Monte-Carlo simulations).

To generate these scenarios, it is necessary to compute the conditional distribution of the remaining number of arrivals of each type. Several methods can be used to evaluate a scenario: we can simulate the behavior of a Best-Fit strategy on this scenario, or the behavior of a far-seeing strategy (exact or greedy approximation), or the behavior of any other "slave" strategy (for instance other strategies described in this paper).

These first experiments tend to reveal that the results of this strategy are not so dependent on the quality of the slave algorithm. For instance, evaluating with a Best-Fit or with a far-seeing algorithm lead to very similar results, whereas the former is a pessimistic evaluation (the final gain is significantly higher in general) and the latter is an optimistic evaluation (it gives an upper bound that cannot be reached). Besides, using an evaluation with the strategy described in the next section does not improve the results, whereas the behavior of this strategy is very similar to that of Forward Sampling. Moreover, forgetting to take into account the possibility for items already present to be cancelled caused a surprisingly small deterioration of results. Finally, since what is important is the sensitivity of this algorithm used for comparisons, a greedy heuristic similar to C1 in Section 4 seems to be a good slave strategy.

On the contrary, Table 1 shows that the number of samples has a great influence on the average gain: even if 8 samples are sufficient to get better results than the Best-Fit (1.67 %), increasing this number up to 1000 leads to an average gain of 8.23%.

Table 1. Influence of the number of samples (1000 runs on the "master" problem)

Number of samples	*8*	*10*	*20*	*50*	*100*	*1000*
average gain (%)	1.67	2.8	6.85	7.89	7.95	8.23

4. Using Expected Values and Combinatorial Optimization (EV)

This second approach is similar to the previous one in its global principle, but uses a precise resolution of a combinatorial problem that is generated from the expected values of the departures/arrival as an oracle of what may happen. To evaluate a current state, we try to compute the mean of the optimal "far-seeing" (a posteriori) filling of the bins. If we designate by C an algorithm to solve this multi-choice knapsack problem, and by C* an optimal algorithm, we can say that we want to use E(C*) as the expected value to guide our decision.

In the previous section we used $\frac{1}{n}\times\sum_{i=1}^{i=n} C(S_i)$ to evaluate E(C*), making use of a simple heuristic C to fill the bins. Here we will use a better algorithm C^+ but we will run it only once, over the mean values of the *a posteriori* problem, yielding $C^+(E(S))$. One can say that the goal of this paper is to evaluate different strategies to produce an estimate of E(C*), and this section deals with the C*(E) approach, where C* is approximated by C+.

This points out to another, more complex but possibly better suited, direction for producing online algorithms. When we evaluate the future, we use an optimal a posteriori strategy, whereas it would be more interesting to use recursively the strategy that is being defined. This type of recursive definition, where the optimal decision at date t is defined using the future decisions at time $t + \delta_i$, may be solved (numerically) using differential equations when the problem is simple enough, but is, to our knowledge, too difficult for this present problem.

However, it is clear that we cannot evaluate the expected penalty for overbooking using an a posteriori strategy: in the *a posteriori* scenario, we decide to simply ignore those reservations that we do not consume for our optimal filling. To take the penalty into account, we add a simple lower bound estimate of the penalty that is expected based on the current balance and the expected cancellation rate. Note that this lower bound is only "exact" if we were to refuse all future incoming reservations, which shows that there is an opportunity for a better analysis and a more precise algorithm.

Thus, we can summarize the online algorithm as follows:

1. Compute expected reservations and reservations from date to deadline
2. Produce average deadline scenario S and compute reference value *V* = C+(S)
3. For each bin *b*,
   compute *v* = C+(S) + ExpectedPenalty(current state) with *item* added to *b*
   pick *b* that maximizes *v* using balance(*b*) to break ties
4. If v >= V, accept *item* and place it into bin *b*, otherwise refuse *item*

As noticed in the previous section, the algorithm is not very sensitive to the quality of the cancellation prediction, although it is important to use a conservative estimate.

The combinatorial algorithm is a constraint propagation algorithm that uses a limited form of branching with a LDS scheme [HG95]. The branching strategy is to pick which kind of items should be inserted into the current solution and then to branch on all possible bins. To evaluate which bin to pick, we measure the difference in the maximal expected value for the bin before and after the insertion. This maximal value is the solution of the single knapsack problem for the bin, applied to all incoming reservation. We use a LDS scheme, and decide to branch on which bin to pick when there are two bins for which the difference between the two valuations is small enough. The single knapsack is solved with another LDS algorithm that is very similar but much simpler since the value of an insertion is simply the current value of the bin.

The next table addresses the importance of the quality of the C+ algorithm. We give here the results obtained respectively with a naïve greedy heuristic (C1), with a smarter algorithm with one level of branching (C2) and last with our complete algorithm presented here (C+). In this table, we have included the standard deviation to substantiate our claims about the statistical relevance of the experiment, thus each cell in the table is a tuple (average, standard deviation).

Table 2. Comparison of C1,C2 and C+

Problem	*BF*	*C1*	*C2*	*C+*
Master	454, 27.7	481, 32.9	484, 33.6	487, 34.8
Pb1	454, 29.1	489, 30.2	492, 31.6	494, 32.7
Pb2	469, 28.9	496, 34.5	499, 34.0	504, 34.8
Pb3	431, 47.4	448, 49.4,	444, 34.9	449, 49.9
Pb4	82, 18.1	83, 17.6	83, 17.6,	83, 17.6
Pb5	453, 29.9	481, 34.1	484, 34.9	487, 35.6
Pb6	11782, 797.6	12852, 1023.7	12935, 1024.3	12951, 1034.6
Pb7	938, 51.5	906, 53.4	922, 56.2	938, 51.4
Pb8	458, 26.5	519, 29.6	528, 30.5	538, 30.4
Pb9	421, 48.5	418, 50.4	419, 49.1	420, 48.7

A more traditional approach in the field of stochastic programming is to concentrate on whether a new request should be accepted, using a stochastic evaluation, and then simply use a best-fit algorithm to pick which bin must be used. This is for instance the case for most “Yield Management” approaches, including the one that will be presented in Section 6. Therefore, we have conducted the following experiment, where the decision of which bin to pick is left to a best–first heuristic, but we use the same analysis (C+(E) with or without the new reservation) to decide whether to accept or refuse the reservation.

Table 3. Full exploration vs. binary strategy

Problem	*Full exploration*	*Binary (simpler) strategy*
Master	487, 34.8	481, 39.5
Pb 1	494, 32.7	433, 64.1
Pb 2	504, 34.8	493, 39.5,
Pb 3	449, 49.9	388, 60.2

One may see that the impact of the combinatorial choice between the bins depends on the type of problem. This table is useful when comparing with results obtained with our preliminary implementation of the Yield Management approach in Section 6, since it is similar to the binary approach presented here.

Although this approach presented in this section produces satisfactory results, it still ignores the true stochastic nature of the problem. To remedy this drawback, there are two possible directions:

1. Develop techniques borrowed from stochastic optimization to better characterize E(C). This is the approach that we plan to investigate in the future. Because the algorithm C+ is fairly complex and uses various heuristics, it may actually be easier to characterize C*, that is the optimal resolution of the combinatorial problem and derive information that also applies to C+.
2. Characterize the stochastic nature of the problem, either with a segmentation, which will be developed in the next section, or through cutting rules, which will be shown at the end of Section 6.

5. Combinatorial Analysis (CA)

5.1. Related Markov Decision Processes

MDP is the model of choice [H60, P94] to build optimal policies for stochastic decision problems. Under the assumptions that the arrival laws (In_k) and departure laws (q_k) are time-independent and non correlated, the on-line Multi-Choice Knapsack Problem can be modelled as a fully observable MDP, with an obviously large number of states.

Classic techniques for solving MDPs like Linear Programming [D63], policy iteration [H60] or value iteration [B57] are not suitable in the case of large state space and advocate the development of MDP-decomposition methods. Various approaches have been proposed for the latter, among which: adaptative aggregation [BC89], Dantzig-Wolfe Decomposition [KC74] or, for planning problems, [BDG95, DL95, BT96]. More recently, new decomposition schemes have been developed that exploit the "weakly coupled structure" of some MDP that occur in online resource allocation: policy caches [P98], dynamic MDP merging [SC98] or Markov Task Decomposition [MHK+98].

Our approach consists in a crude implementation of an event-aggregation strategy, focussing on regions of final states of the MDP.

5.2. Combinatorial sampling

A particularity of our stochastic optimization problem relies in the fact that the revenue is evaluated at the very end of the process. In a sense, the ordering of the incoming events has little importance as no immediate reward or holding cost influence the revenue, provided that the strategy is efficient. The method in this Section takes advantage of this context and is based upon the following property holding for the deterministic case: at time t, if all external events (incoming items, departing items) are known in advance, an optimal strategy consists in choosing for the item the bin that maximizes the revenue of the Multi-Choice Deterministic Knapsack Problem for which:

- All already accepted items that will not be cancelled are forced into the solution.

- All incoming items that will not be cancelled are possible candidates for filling the bins.

Though NP-hard, the afore-mentioned combinatorial optimization problem can be solved with on-the-shelf MIP tools (see [MT90] for a deep description of knapsack-like problems). Let $F^*(P[n,k],j,t)$ denote the optimal Deterministic Knapsack value over all possible choices for an item of type j arriving at t. We refer the reader to [WB92] or [PRK95] for discrete and continuous markovian models related to on-line knapsack and perishable asset revenue management problems. The solution produced by the MIP is the best (in terms of revenue) valid "state" that is reachable from the current state of the system taking corresponding decisions of acceptations and reject.

For the non-deterministic case, our evaluation strategy would ideally aim at enumerating all possible event sets, compute their occurrence probability together with the Deterministic Knapsack revenue, hence deduce the overall expected value of each possible choice of bin for item j: $E[F^*]$.

Not surprisingly, an obvious drawback of this strategy is its computational time. In order to estimate the revenue of each possible final state of the process that are reachable from $P[n,k]$ taking into account the choice at stake, one has to generate all possible pairs $((n^T_1,\ldots,n^T_k),P^T(n,k))$ where:

- N^T_i is the number of items of type i arrived after t and remaining at T if accepted
- $P^T(n,k)$ is the number of remaining items of type k, already present in bin n before t and still present in bin n at T.

with a computational cost of $O(P^{kn} \cdot (D/K)^k)$, if P denotes the maximum number of items of the same type present in the bins at t, and D the maximum number of items that can enter the system between t and T. Thus we limited our final states study to the most probable states, above a given threshold, restraining ourselves to typically 10000 states at all. Finally, since evaluating each final state by computing the best MIP solution can potentially require a couple of minutes, we use the C1 heuristic instead (described in section 4).

The experimental results not surprisingly indicate that this "combinatorial" sampling offer results that are close to the forward sampling strategy of section 3.

6. Yield Management (YM)

Yield Management techniques analytically estimate impact on the expected final revenue. As with thresholds principle [GR99], at each new arrival is computed expected final revenue and the item is accepted if its marginal revenue improved it.

In a first step, we will describe the computation of the final total balance (FTB) and the final total revenue ($\tilde{F}$) in a deterministic vision. In a second step, we extend these notions for the stochastic case (probability and penalties) and deduce the stochastic expected final revenue with penalties (ϕ).

Finally, we observe that YM results are better than those of the naïve Best-Fit (BF) strategy in most of the cases. Limitations are two fold: multi choice blindness (the multi bin aspect wasn't involved in the formulas) and packing blindness (the major limitation). The influence of this last combinatorial dimension is illustrated on some special instances.

6.1. Deterministic case

Let us start with the simple relaxation defined by using only one bin with capacity C.N and one demand type with size $\overline{w}$ (average of real sizes). Since in this case, maximizing the revenue means maximizing the balance, we compute the Final Total Balance (FTB) witch is the sum of all arrivals. As total arrival exceeds C.N, the first intuition, to remove overloading penalty, is to adjust it with a p_{accept} ratio. We have that $\left(\overline{w}\sum_{k=0}^{K} In^k(1-Out_{In}^k)\right).p_{accept}$ is equal to exactly C.N when refusing arrivals with a $(1-p_{accept})$ probability. A way to increase revenue, instead of refusing every item with a fixed probability, consists in applying price *segregation* whilst refusing only low price items and accepting good ones. In order to compute a threshold between *low* price and *good* price, we sort items types (re-numbered $\tilde{k}$) by decreasing marginal revenue ($v_{\tilde{k}}/w_{\tilde{k}}$) and incrementally compute the FTB by aggregating all accepted items. We find $\tilde{k}*$ such that $\tilde{B}_{\tilde{k}*}=\sum_{i=1}^{\tilde{k}*} In_i(1-Out_{In_i,i})w_i$ just exceeds C.N. For this item type, we apply a p_{accept} ratio. Thus, expected revenue become:

$$\tilde{F}=\sum_{k=1}^{\tilde{k}*-1} v_k In_k(1-Out_{In_k,k})+v_{\tilde{k}*}In_{\tilde{k}*}(1-Out_{In_{\tilde{k}*},\tilde{k}*})\left(\frac{C.N-\tilde{B}_{\tilde{k}*-1}}{\tilde{B}_{\tilde{k}*}-\tilde{B}_{\tilde{k}*-1}}\right) \quad (2)$$

6.2. Stochastic case

To take into account the distribution of possible arrivals, let us focus now on the computation of the final revenue, which is a combination of expected stochastic revenue (ER) decremented by expected stochastic penalty (EP). This stochastic revenue $\phi = ER - EP$ is computed with the aggregated request volume, on all the items that will not be cancelled: $p_{In}(x)$.

$$\phi=\left(\frac{\overline{v}}{\overline{w}}\sum_{x=0}^{\infty} p_{In}(x)[x.\overline{w}]\right)-\alpha\left(\sum_{x=\left\lfloor\frac{CN}{\overline{w}}\right\rfloor}^{\infty} p_{In}(x)[x.\overline{w}-C.N]\right) \quad (3)$$

To understand the threshold idea, we can describe graphically ϕ as shown on next figure according to the level of arrival still controlled by our p_{accept} ratio. In the beginning of the dashed line, the revenue for each new accepted arrival does not yield a penalty probability. There is a linear progression of revenue depending on amount of arrivals, with a rate a_R. At a certain level C_1, a small part of the arrival distribution overloads the capacity C.N. The cost function penalty (EP) starts to reduce the amount of additional revenue in ϕ. At level C_2, all the additional arrivals become penalty. In this last portion, we have a linear behavior for the revenue with rate a_R-αa_P. Now, in the full line, if we cumulate the price segregation with this stochastic estimation of revenue, we apply the same strategy on incremental computation based on ϕ by accepting every item until the final expected revenue decreases: this is the desired

threshold item $\tilde{k}^*$ with his threshold revenue r^*. As illustrated on the next figure (where a_{ri} is the ith best marginal revenue), the price segregation offers a better revenue than the one-item vision.

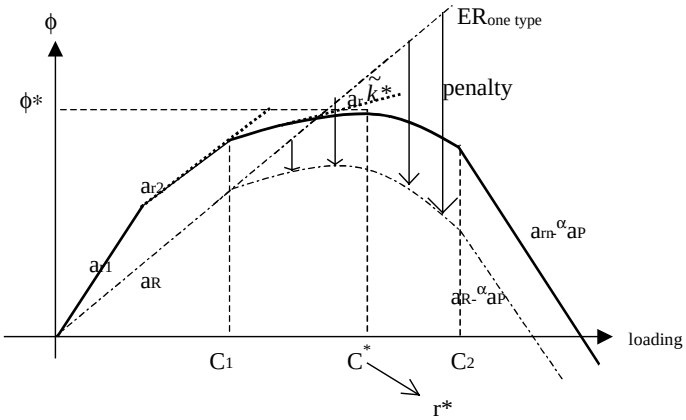

Fig. 2. Revenue as a function of the current balance

As described in the introduction of this part, we dynamically recompute the r* threshold values to decide if we accept arrivals (hence the p_{accept} ratio is no longer needed).

6.3. Limitations and extensions.

The YM strategy results presented in the next section exhibit a better behavior than BF on almost all instances. The disappointing results (compared to other approaches) illustrate clearly the weakness of our current approach from a combinatorial perspective (for instance, on pb3 which packs at most 2 items by bin).

In order to illustrate this last limitation, we have extended the master characteristics by only increasing the average number of item in a bin. Clearly the combinatorial packing aspect (master benchmark: at most 5 items by bin) is smoothed when this number increased.

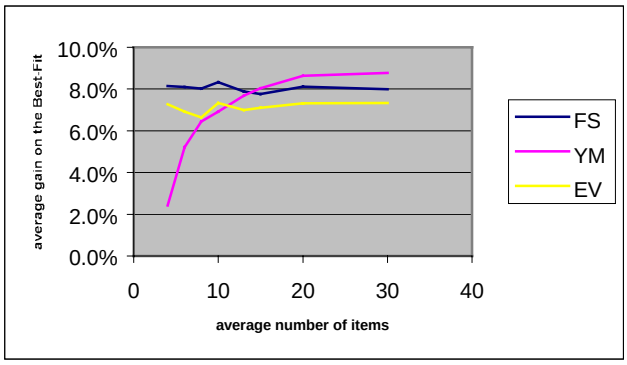

Fig. 3. Influence of the average number of item by bin

Although it may appear from this preliminary experiment that the YM approach is too difficult to tune for this type of combinatorial problems, this is not the case. For instance, we may use the YM analysis to generate simple cut-off rules that will prevent from accepting the lowest value/size ratio during the X first periods. Embedding these rules to EV (section 4) leads to a revenue 7.93% better than BF on the master problem (compared to 7.19% without these rules).

7. Comparisons

This paper is clearly a first step towards an ambitious goal and, therefore, suffers from many limitations. For instance, we plan to extend the experience to stochastic sizes of groups (with an associated probability distribution). Last, we have limited our first experiments to a size that is well suited to analysis, but larger experiments are also required.

7.1. Results

The following table compares the efficiency of our four algorithms: FS (with 1000 samples), EV, CA and YM (respectively in sections 3,4,5 and 6). BF designs the Best-Fit and XMP the optimal far-seeing method quoted in 2.4 (→competitive ratios on the master: FS=90.6% vs. BF=83.8%).

These results are the average results on 1000 runs so that the half width of the confidence interval at 95% is typically 0.7 %. The final column gives the average gain of the best algorithm (bold underlined results) against the BF.

Table 4.Comparative results

Problem	Comment	BF	FS	EV	CA	YM	best gain (%)	*XMP*
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 =$10size^{1/2}$ (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	Run time (s) for 1 scenario master	0.01	20	0.5	60	0.02		

One may notice that run-times vary considerably from one approach to the other. We designed these algorithms to be usable in a human-interaction context, with a response time of around one second (thus 60s for a complete iteration is acceptable). Obviously some on-line problem would require a faster approach such as YM or EV.

Although this is a preliminary report, these results *show the interest of mixing insights from the stochastic and the combinatorial optimization field*. For instance, we showed that using a better CO algorithm in the EV method pays off. On the other hand, there is balance to be found and using a sophisticated optimization technique is easily wasted if the level of stochastic analysis is too low. Another illustration of our central claim is that YM techniques used without using combinatorial insights do not seem competitive, but for "fluid" instances (cf. 6.3), whereas YM filters may improve significantly a combinatorial method.

7.2. Robustness

In all scenarios tested in 7.1, events are generated according to the distributions that are known to the algorithms. On the contrary, this section evaluates the robustness of our approaches, when based on a slightly erroneous estimation of the input distribution (as in real life): wrong law (Table 5) or wrong average (Table 6).

Table 5.Non-regular distributions

Problem	Comment	BF	FS	EV	CA	YM	best gain (%)
Pb29	non-regular distribution (avg:8.08409, stdev:1.87479)	454	**492**	484	490	463	8.2
Pb30	"Very" non-regular distribution (avg:8.01173, stdev:3.12526)	453	444	454	-	**457**	0.9

Table 6. Influence of estimation errors on the input distribution

estimation error in %	*-25*	*-12.5*	*0*	*+12.5*	*+25*
average gain of FS (%)	3.83	**6.85**	**8.23**	**8.45**	6.93
average gain of EV (%)	**6.15**	6.63	6.34	7.40	**7.71**
average gain of YM (%)	**-8**	-3.7	2.4	1	**-1.7**

Facing non-regular distributions, our strategies become very sensible to the standard deviation (especially CA). Estimation errors on the input average have less influence (except for YM) but in both cases, the most robust behavior consists in focusing on average values.

8. Conclusions

We have presented different algorithms that try to solve an online combinatorial optimization problem through different techniques that borrow from different fields. Although we report preliminary results, they are stable enough to confirm the need for

combining insights from combinatorial and stochastic programming. We argue that constraint programming is a relevant technique for such problems for two reasons:

1. On the one hand, many problems for which CP is well suited because of its expressive power (its ability to capture situations that are difficult to model) have a stochastic nature
2. On the other hand, two of the algorithms presented here can be easily adapted to a CP approach for a given problem: the simulation approach (FS) and the expected value approach (EV)

We have identified different topics that must be investigated before we may develop a generic method for solving stochastic combinatorial optimization problems with constraints:

1. How does one obtain stochastic indicators about a lower or upper bound that is computed by a (limited) branch-and-bound algorithm?
2. How can an abstract search space be abstracted from a stochastic description, onto which a combinatorial approach may be found?
3. How does one develop fast, incremental CP simulation engines, which is a request that is also found when developing hybrid methods that combine CP and metaheuristics such as stochastic methods?

In this last question, we see that the combination of CP and stochastic has been mostly studied so far with the angle of introducing stochastic methods (such as randomized algorithms) as opposed to solving stochastic problems. For instance, one may notice the difference with the proposal of Probabilistic Constraint Programming (PCP [DW00]), where probabilities are assigned to constraints as opposed to data. However, we plan to investigate how PCP could be used to solve this type of online problems. Last, we want to emphasize the availability of our benchmark problem (www.elab.bouygues.com/adhoc/prototypes/stokp), which we hope to see attempted by other teams with other methods.

9. References

[ABC+99] F. Afrati, E. Bampis, C. Chekuri, D. Karger, C. Kenyon, S. Khanna, I. Milis, M. Queyranne, M. Skutella, C. Stein, and M. Sviridenko: *Approximation schemes for minimizing average weighted completion time with release dates.* Proc. of the 1999 Symposium on Foundations of Computer Science (FOCS99), 1999.

[AW96] A.N. Avramidis, J. R. Wilson : *Integrated variance reduction strategies for simulation*. Operations Research 44 (2): 327-346, 1996.

[B57] R. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[BC89] D. P. Bertsekas, D.A. Castanon. *Adaptative aggregation for infinite horizon dynamic programming*. IEEE Transactions on Automatic Control, 34(6):589-598, 1989.

[BDG95] C. Boutilier, R. Dearden, M. Goldszmidt: *Exploiting structure in policy construction.* In Proceedings of the 1995 international joint conference on artificial intelligence, 1995.

[BL 97] J. Birge, F. Louveaux,: *Introduction to Stochastic Programming*
Springer Series in Operations Research, 1997.

[BT96] D. P. Bertsekas, J. N. Tsitsiklis: *Neuro-dynamic Programming*. Athena, Belmont, MA, 1996.

[CSL99] Y. Caseau, G. Silverstein. F. Laburthe, *A Meta-Heuristic Factory for Vehicle Routing Problems*, Proc. of the 5th Int. Conference on Principles and Practice of Constraint Programming CP'99, LNCS 1713, Springer, 1999.

[D63] F. D'Epenoux. *A probabilistic production and inventory problem*. Management Science: 10:98-108, 1963.

[DL95] T. Dean, S. H. Lin. *Decomposition techniques for planning in stochastic domains*, In Proceedings of the 1995 international joint conference on artificial intelligence, 1995.

[DW00] A. Di Pierro, H. Wiklicky: *Randomised Algorithms and Probabilistic Constraint Programming* Proc. of the ERCIM/Compulog Workshop on Constraints, 19-21 June, Padova, Italy, 2000.

[FW97] A. Fiat, G.J. Woeginger: *Online Algorithms* Lecture Notes in Computer Science, vol. 1442.

[GR99] J.I. McGuill, G.J. Van Ryzin *Revenue Management: research overview and prospects* Transportation science vol.33 n°2, may 99.

[H60] R.A.Howard: *Dynamic programming and markov Chains*. MIT Press. Cambridge, 1960.

[HG95] W. Harvey, M. Ginsberg : *Limited Discrepancy Search*. Proceedings of the 14th IJCAI, p. 607-615, Morgan Kaufmann, 1995.

[HSS+96] L. Hall, A. S. Schulz, D. Shmoys, J. Wein: *Scheduling to Minimize Average Completion Time: Off-line and On-line Approximation Algorithms.* Proc of SODA: ACM-SIAM Symposium on Discrete Algorithms, 1996.

[KC74] H. J. Kushner, C. H. Chen: *Decomposition of systems governed by Markov chains.* IEEE transactions on Automatic Control, AC-19(5):501-507, 1974.

[MHK+98] N. Meuleau, M. Hauskrecht, K.-E. Kim, L. Peshkin, L.P. Kaelbling, T. dean, C. Boutilier. *Solving Very Large Weakly Coupled Markov Decision Processes.* American Association for Artificial Intelligence, 1998.

[MT90] S. Martello, P. Toth, *Knapsack problems. Algorithms and computer implementations.* John Wiley and Sons, West Sussex, England, 1990.

[P98] R. Parr. *Flexible Decompostion Algorithms for weakly coupled Markov Decision Problems.* Uncertainty in Artificial Intelligence. Madison, Wisconsin, USA, 1998

[P94] M. L. Puterman. Markov Decision Processes: *Discrete Stochastic Dynamic Programming*. Wiley, New York, 1994.

[PRK95] J. Papastavrou, S. Rajagopalan, A. Kleywegt : *The Dynamic and Stochastic Knapsack Problem with Deadlines*. Technical report, School of Industrial Engineering, Purdue University, West Lafayette, April 1995.

[SC98] S. P. Singh, D. Cohn. *How to dynamically merge Markov Decision Processes.* In M. Mozer, M. Jordan and T. Petsche eds, NIPS-11. MIT Press, Cambridge, 1998.

[WB92] L.R. Weatherford, S.E. Bodily: *A Taxonomy and Research Overview of Perishable-Asset Revenue Managment : Yield Managment, Overbooking and Pricing*. Operations Research 40:831-844, 1992