Performance testing of a predictive dialling algorithm for Computer Telephony Integration systems

LUIGINO BENETAZZO[*], MARCO CORTESE[**], RICCARDO COSTACURTA[**]
(*) Department of Electronic and Computer Science - University of Padova
Via Gradenigo 6/a, ITALY

(**) Deimos Italia s.r.l. - Padova Via Pellizzo 14/E, ITALY

Abstract: - The Call Center world is in continuous expansion, and Computer Telephony Integration technologies (CTI) can introduce significant improvements. Call Centers are generally used to accept inbound calls and provide services with human or automatic agents. An additional application field is emerging and attracting business attention: outbound systems. Using the same infrastructure, a Call Center can be used to place calls and reach customers, for example, for telemarketing activities. CTI allows to automate and optimize many tasks, but performance and productivity mainly depend on the dialling algorithm: in this paper predictive solutions will be considered, also proposing a new approach. Its implementation features will be studied, outlining some typical performance testing problems. Particular concern will be also placed on small centers where a Markov queue model generally can't obtain acceptable results.

Key-Words: - Call Center, Predictive dialling, Outbound Systems, Computer Telephony Integration.

1 Introduction to Outbound dialling systems optimization

The Call Center world seems to be the richest context for Computer Telephony Integration (CTI) based solutions. Call Centers are by now used as service centers, in which technology allows to manage incoming phone calls and assign them to skilled human agents or automated Interactive Voice Response (IVR) devices. However another new interesting application field is outbound dialling, that is, when the system places outgoing calls and assigns only answered ones to agents (telemarketing, customer care, fund rising, etc...). In outbound applications, CTI technology [1] allows to automate call placing activities (*launching or dialling engine*), recognize if a human customer is answering and then transfer calls to the phoneset of a target agent (human operator). In this context human agents are valuable system resources and their productivity depends on the total amount of time spent on conversation or data processing (*working time*) and on their *availability* (that is, when they are logged-in and set ready). Calls that receive a human answer will be followed by a conversation with the agent (*live calls or contacts*). From the system point of view this can be thought of as a hit result that will impact on system performances. There are many different algorithms and approaches to implement these jobs, but all of them have the same target: the optimization of call center performances, that is, maximizing agent productivity while introducing minimum nuisance to customers. Different solutions have been studied: preview dialling, power dialling, progressive dialling and predictive dialling [2,3]. All but the first are based on the concept that placed calls have an intrinsic death rate (busy destination, no answer, fax, answer machine, etc...) and as a consequence only live calls must participate to agent assignment tasks. This is the basic reason why these algorithms introduce an overdialling rate, that is they place more calls than available agents. When a customer answers, if the system has a free agent it executes matching, otherwise it can't manage the contact: these calls are called *Nuisance Calls*. Predictive dialling systems are characterized by the possibility to manage overdialling on a statistical basis, keeping nuisance level under control. Different algorithms can fulfill to these requirements.

Most predictive dialling solutions [3,4] have a decision strategy based on a Markov queuing model that computes in advance a launching pace and then places calls at this rate. The main job is thus performed off-line, computing statistical indices to describe customers behavior, according to a queue model [5,6]. This approach generally leads to suboptimal performances, especially in low sized systems (i.e. those containing only a few agents).

In this paper an innovative approach is proposed, based on statistical analysis accomplished at run-time over significant intervals, rather than on stochastic intensity parameters evaluated off-line and that describe the whole system. A feedback has also been added in order to improve algorithm convergence and control output parameters. Algorithm robustness and accuracy is also studied by means of a comparative

analysis with a solution based on a Markov model, with particular concern on systems with a medium-small number of available agents.
A characterization of statistical algorithms is necessary because the proposed solution is based on system performance measures and on the consequent tuning of the dialling engine to desired preset targets.
Adopted models are actually based on statistical descriptions (s.d.) of random variables (r.v.) representing quantities (state sojourning times) involved in the system. Also stochastic processes (s.p.) are used to describe events and their intensity (calls placed, calls answered, etc…). It is supposed that measure sequences are available to the launching engine, obtained in a non intrusive fashion and that such s.d. are computed with a sufficient accuracy [2].

2 Statistical definition of measure parameters

From a functional point of view, a predictive dialling system can be schematized as in Figure 1, where the dialling engine can be described by a statistical launching intensity λ; $p(t)$ is the probability that communication is set (*Hit Rate*), modeling customer behavior; Φ is the function describing dynamic system behavior during assignment of contacts to agents and their statistical state evolution, producing actual live contacts or unmanageable calls. *NCR(t)* and *Pr(t)* describe system performances in terms of nuisance calls and agent productivity and will be now defined. Optimization and algorithm choices thus concentrate on the launching engine, and can be built on different mathematical models (off-line Markov one, greedy on-line, etc...).
The main entities that should be measured are listed below and are involved both in Markov model algorithms and in the proposed one.

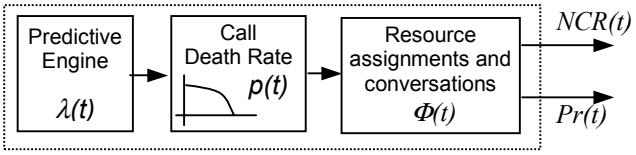

Fig. 1: Block diagram of an outbound dialling system with a predictive engine.

Hit Rate (p(t)): represents the ratio of calls terminated in a phone contact with respect to the total amount of placed ones. In the statistical model it defines the probability of a human answer: this implies that $0 \le p(t) \le 1$. Its evolution should be measured in order to use it in further estimation activities. This quantity is also strictly time dependent so it should be computed by putting together values obtained by measurement sequences on a moving time window. This window should be defined so that obtained instantaneous values of $p(t)$ can be considered meaningful. A value of about 30 seconds has experimentally provided good results and exhibits the further property of being much shorter than application time duration: outbound activities, called *campaigns*, generally have a mean duration of about 3-4 hours or more. At the same time this interval has the same magnitude order as the time needed by a call to reach a contact and thus evaluate its hit result.
Nuisance Call Rate (NCR(t)): is the ratio of live contact calls with no agents available on the total amount of placed ones, from the beginning of a campaign up to time t. It's supposed that these calls are immediately hung up by the system, as actually happens in most existing solutions. $NCR(t)$ is a cumulative quantity and system performances can be deduced by its final value $NCR(t_F)$ where t_F is the time of campaign end, possibly compared with a preset target value NCR_0. Its measurement is also necessary because its instantaneous values are used in feedback and self adaptation phases, as will be shown in Section 3 below. Moreover, a time evolution analysis during the campaign can be useful to evaluate algorithm convergence and stability.
Agent productivity (Pr(t)): it is the main optimization target of predictive dialling systems, defined as the ratio between active working time of all agents and their available time at time t. It can be described as follows:

$$Pr(t) = \frac{\sum_{i=1}^{n_c(t)} t_a(i)}{\sum_{i=1}^{n_c(t)} t_w(i)}$$

where $n_c(t)$ is the number of managed live calls until time t, $t_a(i)$ is the i-th contact duration and $t_w(i)$ is the measured waiting time for the agent to get the $(i+1)$-th contact assigned by the system. $Pr(t)$ is a cumulative quantity, so as time flows its oscillations are smoother (see Figure 4). Its instantaneous value is used for estimation and feedback, but in order to evaluate algorithm effectiveness it must be evaluated in its final value for $t=t_F$. In this case, too, its time evolution shows algorithm stability and convergence indices.
Launching rate ($\lambda(t)$): it is an intensity parameter describing the number of system dialled calls per time unit; its statistical value can be the result of different predictive engine algorithms. For instance in the Markov queuing model, where dialling pace (Δt) is a evaluated a priori, it assumes a constant value $\lambda(t)=\lambda=1/\Delta t$. However, as stated earlier, only

answered calls are assigned to agents, so assignment times (arrivals in the model) are non deterministic: call death rate makes the process stochastic, altering its statistical description. It thus become a Poisson s.p. with arrival rate $\lambda'(t)$, where $\lambda'(t)=\lambda(t)\cdot p(t)=p(t)/\Delta t$, that is, call death rate causes the equivalent rate to reduce.

$$\begin{cases} NCR(t) = \Phi_1(\lambda,t) \\ Pr(t) = \Phi_2(\lambda,t) \end{cases} \quad (1)$$

Observing Figure 1, a predictive dialling system can be described by input-output equations reported in (1). The optimization problem can then be rewritten as in (2), where performances must be measured through all campaign duration because of the cumulative definition but performance optimization problem must focus only on their final values; λ is the free parameter, whose meaning depends on adopted model.

$$\begin{cases} min_\lambda = NCR(t_F) \\ max_\lambda = Pr(t_F) \end{cases} \quad (2)$$

Block Φ: represents calls and system evolution; in order to describe it, other statistical quantities play an important role. A finite state machine (FSM) model has been chosen to describe the evolution of calls and agents [2]. Some r.v. will then be used to describe state sojourning times, with the hypothesis that their complete statistical description, i.e. the probability density function (PDF) $f_{t_i}(a)$, is known. Agents and calls states will randomly change from *i* to *j*: this is described by probabilities p_{ij} so that $\sum_j p_{ij} = 1$ and the adopted model has the features of a Markov sequence. Predicted Hit Rate is $p = \prod_{i \in E} p_{ij}$ where *E* is the set containing states leading to a live contact.

3 Metrological parameters of the Algorithmic solution

A Markov queue model approach can be used adopting an *M/G/c/c* model (as usually indicated in literature) and Erlang Beta equation can be easily used to size launching pace. Even if it's widely supported by the literature [5,6], the approach does not seem to lead to the desired results in predictive dialling applications in centers with few agents (up to 10).

Markov solutions seem to have big limitations due to their low flexibility: the launching pace λ is predimensioned depending on the rate λ' which should describe homogeneous stochastic processes, but system events can hardly satisfy this feature. Another limitation comes from the assumption of fixed size queue while agents availability (servers in the model) is strongly varying. In other words, the Chapman-Kolmogorov law of large numbers often can't overcome statistical parameter dynamics ($p(t)$ and $\lambda'(t)$ may change widely with time from the mean values used by the model), leading to suboptimal performances, especially in medium-small sized call centers. Figure 2 displays the high sensitivity of performances when statistical parameters move away from predicted working points [1]. In this example an estimated Hit Rate $p=0.5$ is used to compute launching rate λ in order to get a target NCR_0. If at time *t* some conditions lead to a real value of $p(t)=0.7$, Erlang Beta formula leads to estimate NCR $\approx$ 2 NCR_0 which is generally not acceptable.

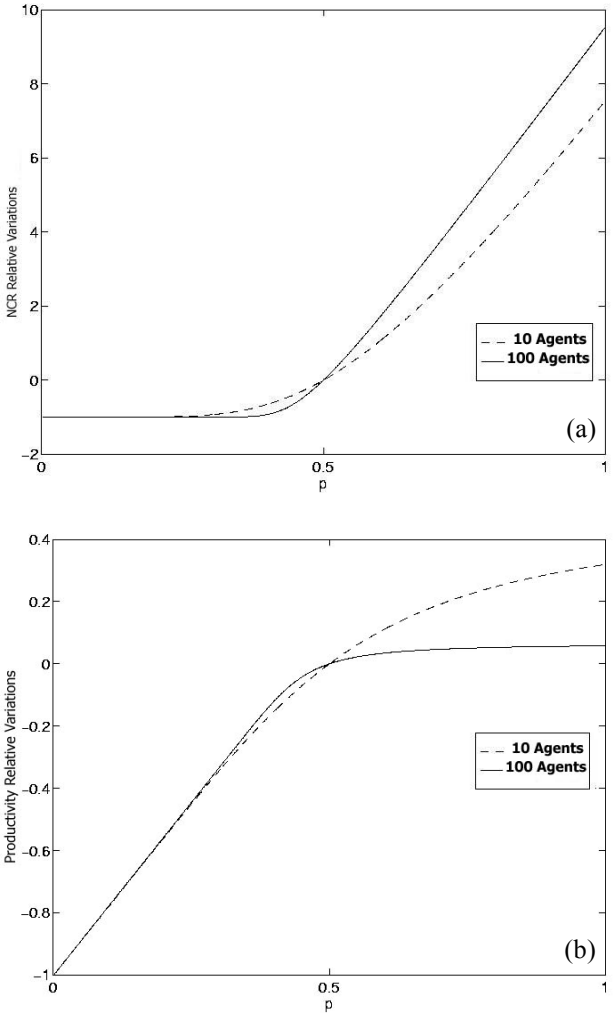

Fig. 2 (a,b): Diagram of performance indices variations on Hit Rate relative variations with respect to its value used in pace presizing ($p=0.5$).

In this context a different approach seems to be more appropriate: the use of run-time working conditions in place of predefined statistically computed quantities. An on-line greedy algorithm [7] has been

studied: it evaluates system state and produces predictions in a limited and reduced time window (T_{win}); then it takes launching choices leading to a local optimum in relation to the same time interval. The choice then moves from a time pace presizing task to a run-time choice of the exact number of calls to place in order to optimize performance quantities in the short period. Since global performances are obtained by aggregating values collected during system evolution, a local optimum reflects on a global one.

For the purposes of a numerical computation, the time axis must be split into time slices (T_C) shorter than the analysis interval and predicted time quantities. Estimation time interval (T_{win}) has been assumed to be the statistical time needed by a call to reach a contact. Statistical resource matching, and thus launching choice, takes place within this time window and is evaluated each T_C seconds.

In the FSM model, random variables are used instead of s.p. to describe agents and calls sojourning times in their states; they can also take advantage of an additional knowledge, that is they are staying in that state since a known amount of time. This allows more accurate computation of the number of statistically available agents ($n_{ag}(t)$) and the number of calls that could require them ($n_{cl}(t)$) within the time window ($t,t+T_{win}$). At time t the system will place as many calls as needed to balance these expected values. In addition to evolution times, it is also important to take into account call death rate, described by $p(t)$ (with a slower dynamics) whose actual description at time $t = n \cdot T_C$ is known. The main idea is that at time t the system should place $n_l(t)$ calls, where:

$$n_l(t) = \frac{n_{ag}(t) - n_{cl}(t)}{p(t)} \qquad (3)$$

Using $T_C \ll T_{win}$ allows to adjust choices made in the previous step, adapting them to the new system state and new s.d. of each r.v., thus improving prediction quality.

Such a system can then be described as in Figure 3, ignoring the feedback block $f(z^{-n_d})$. S represents the prediction and launching algorithm: it computes values $n_{ag}(t)$ and $n_{cl}(t)$ based on available PDFs , implements equation (3) and places $n_l(t)$ calls. Block Φ represents real calls evolution (answer, busy signal, fax answer, lines congestion, etc…) and system management tasks (agent assignment or hang-up) leading to conversations or nuisance calls, that are contributing to the evolution of performance indices. A complete statistical description of Φ depends on interdependent random variables, leading to a very complex and generally unaffordable mathematical treatment. Moreover a joint statistical description for these r.v. is not available, neither is the a posteriori PDF describing output conditional probability with respect to input r.v. (and vice versa): estimate and decision theory can't then be applied. As a consequence estimation is based on a partial statistical description of Φ, so to improve algorithm controllability, a feedback has been introduced.

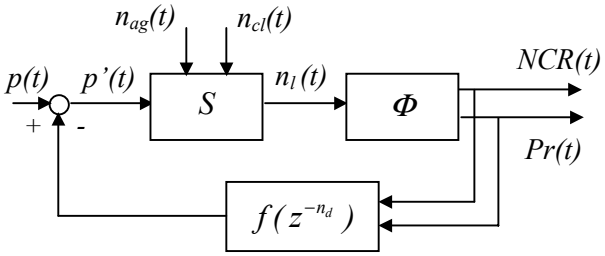

Fig. 3: Schematization of run-time system with a feedback control block.

This allows evaluating the effects of past step choices and then tuning launching decisions according to results. The main idea is to provide a self-learning behavior to function Φ represented in Figure 3, using measured values at the input and output of that block. A greedy algorithm can then take advantage of a tuning feature, correcting limitations due to limited knowledge of statistical description of block Φ.

Feedback time delay must be comparable with the time needed to evaluate any consequence of the choice made, that is with time window T_{win}. So the number of delay steps n_d in the discrete representation of Figure 3 is the number of clock intervals composing this time window, that is, $n_d = T_{win}/T_C$. Feedback is redirected to the input as a Hit Rate correction factor, to be coherent with the assumed model. So $p'(t)$ is composed of a measured component and a correction component based on run-time measured performance parameters and their deviation from preset values. Experimentally, using a controlled feedback model also allows the algorithm to converge faster.

Another interesting point is the possibility to use a much more accurate system evolution time s.d., balancing the loss of accuracy due to the fact that the observation interval is finite (T_{win}) and s.d. are discretized. It is therefore possible to know at each time t, for each call and agent, since how many seconds $t_{s_{i_0}}$ it is staying in the current state; these s.d. are more accurate, so it's useful to take into account their conditional PDF. They are easily obtained by their corresponding unconditional PDF (on which the Markov model is based). PDF $f_{t_{s_i}}(a \mid a > t_{s_{i_0}})$ is

obtained from $f_{t_{s_i}}(a)$, thus conditional expectation is:

$$E\left[t_{s_i} \mid t_{s_i} > t_{s_{i0}}\right] = \frac{\int_{t_{s_{i0}}}^{\infty} t' f_{s_i}(t') \cdot dt'}{\int_{t_{s_{i0}}}^{\infty} f_{t_{s_i}}(t') \cdot dt'} \qquad (4)$$

Conditional PDF are narrower, and their power spectrum is concentrated in a narrow prediction interval, leading to more useful expectations values.
Statistical descriptions mentioned earlier are obviously computed from previously measured sequences: they are generally called "historical" statistics. It is however possible to use a run-time sampled counterpart, computing the corresponding PDF, that is, having an actual statistical description of agents and calls at run-time. It is also useful to adopt a weighted balance of these two conditional expectation samples that may be strongly different. This is another positive contribution of on-line greedy algorithms.
Numerical representation of these r.v., which are continuous in the model, may lead to theoretically unpredictable quantization errors [8]. In [2] a method has been proposed, allowing vector computation and update algorithm complexity to be a linear function of *N*, where *N* depends on the required sampling resolution. This may be useful in real-time environments such as telephony.
The proposed algorithm also introduced a quantization problem due to algorithm model rather than to its numerical computation. Launching equation (3) is purely theoretical because $n_l \in \mathbb{N}$, so it must be rewritten as

$$n_l(t) = \rho\left(\frac{n_{ag}(t) - n_{cl}(t)}{p(t)}\right) \qquad (5)$$

where ρ is the round function. A study of algorithm sensitivity may be useful because the bigger it is, the faster the system reacts to changed working conditions. Equation 5 has a small sensitivity on its parameters: starting from a working point (n_{ag} ,n_{cl} ,p), with other conditions unchanged between t and $t+T_C$, launching decision will change when $\left|n_l(t+T_C) - n_l(t)\right| = 1$ and substituting into equation (5) it's possible to deduce that it will occur when, in absolute value, the percent changes of *p(t)*, n_{ag} or n_{cl} satisfy the conditions reported in (6).
Sensitivity against *p(t)* may reach values near 20% in low sized centers, but will move down rapidly toward 1% as the number of available agents increases. So this algorithm has a low sensitivity to variations with time of *p(t)*, which however has a slow dynamic if compared to other parameters. Moreover feedback action reported on *p(t)* (see Figure 3) leads to a higher algorithm convergence speed [2], keeping it near 20 minutes.

$$\begin{cases} \left|\frac{\Delta p}{p}\right| = \left|\frac{p(t+T_C) - p(t)}{p}\right| = \frac{p(t+T_C)}{n_{ag} - n_{cl}} \\ \left|\frac{\Delta n_{ag}}{n_{ag}}\right| = \left|\frac{n_{ag}(t+T_C) - n_{ag}(t)}{n_{ag}}\right| = \frac{p}{n_{ag}(t)} \\ \left|\frac{\Delta n_{cl}}{n_{cl}}\right| = \left|\frac{n_{cl}(t+T_C) - n_{cl}(t)}{n_{cl}}\right| = \frac{p}{n_{cl}(t)} \end{cases} \qquad (6)$$

The two other equations in (6) evidence their linear dependence on *p,* and since they are integer values, they are surely greater than *p,* so their effect is immediate.

4 Experimental results

Experimental measurements of the proposed on-line algorithm performances emphasize some interesting features. Figure 4 shows how the algorithm is intrinsically convergent, even when the feedback stabilization effect is not present. Tests show a good algorithm adaptivity to working condition changes, taking performances back to desired values (see Figure 5).

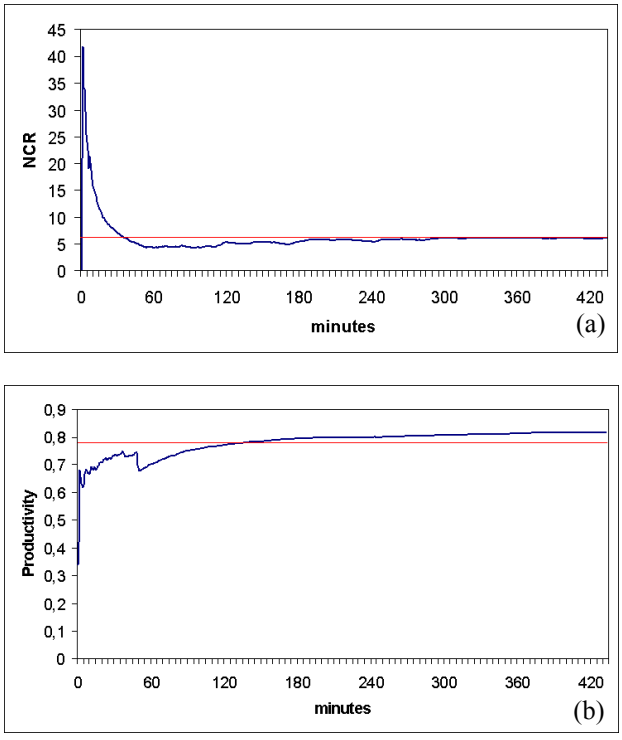

Fig. 4 (a,b): Performance indices evolution with no feedback block in a real situation.

They have also indicated that performances improve as system size grows (as stated in previous sections) but at the same time they do not significantly degrade when system size reduces. The values of *NCR* obtained in more than 50 tests, in different working conditions and with a very strict preset target NCR_0 (1%), never exceeded 2%. Productivity level was always over 60% (36 min/h) for small centers and reached 80% (48 min/h) in medium sized centers (20 agents). An example is reported in Figure 5 where it's visible how indices need about 1 hour to reach statistically significant values: this means that all agents are then continuously working. At this time *Pr(t)* comes to a steady value which is maintained until the end without relevant variations. *NCR(t)* shows some oscillations due to the change of working conditions that lead to a too high overdialling rate; the immediate feedback effect is also visible: nuisance level is then kept under control.

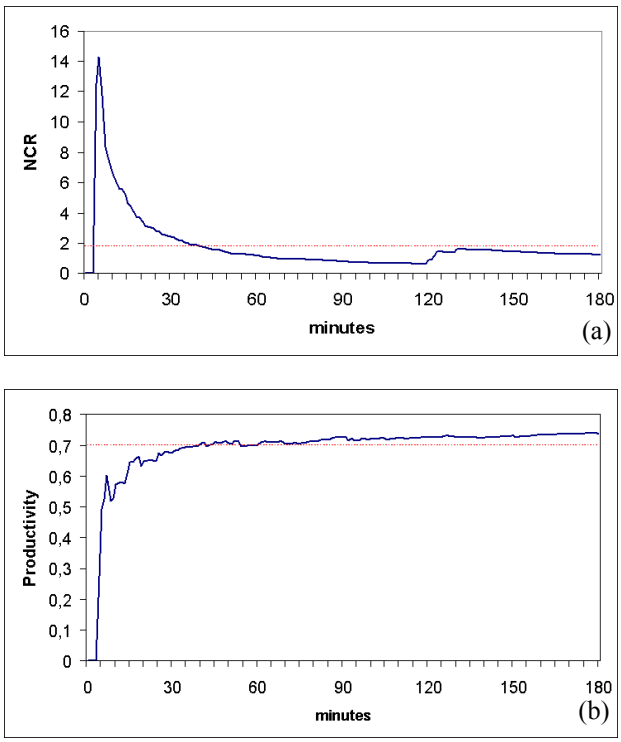

Fig. 5 (a,b): Real test diagram of measured performances in a 5 agents system with a predicted Hit Rate equal to 60%.

This doesn't affect agent productivity, and experimental results confirm how in the long distance also *NCR(t)* is convergent. These parameters are cumulative and they are less sensitive to variations of their instantaneous values as time flows, so oscillations get smoother. This behavior outlines that in order to get good performances, campaign duration can't be too short and at the same time that these results can be considered as lower limits. All results have been obtained in 3 hours campaigns, whose duration is comparable with real ones. As a consequence campaign duration seems to be long enough to reach meaningful steady values. Comparing performance measures with Markov theory based algorithms with a predefined pace, the most interesting results concern small centers, with at most 10 active agents. Some of these results are outlined in Figure 6. The x-coordinate represents the stimulation *pattern,* that is, conditions leading to equivalent Hit Rate variations from predicted value (38%) up to 50%.

For *NCR* results it's quite evident that off-line pace precomputing is too strict and rigid, leading to sub-optimal results if compared with desired targets. The on-line solution instead is always near the planned limit, trying to get the best from available resources. Productivity measurements outline how an off-line algorithm can't go over 40%, while the proposed one never falls under 60%. It also keeps performances quite constant when working conditions move away from predicted ones.

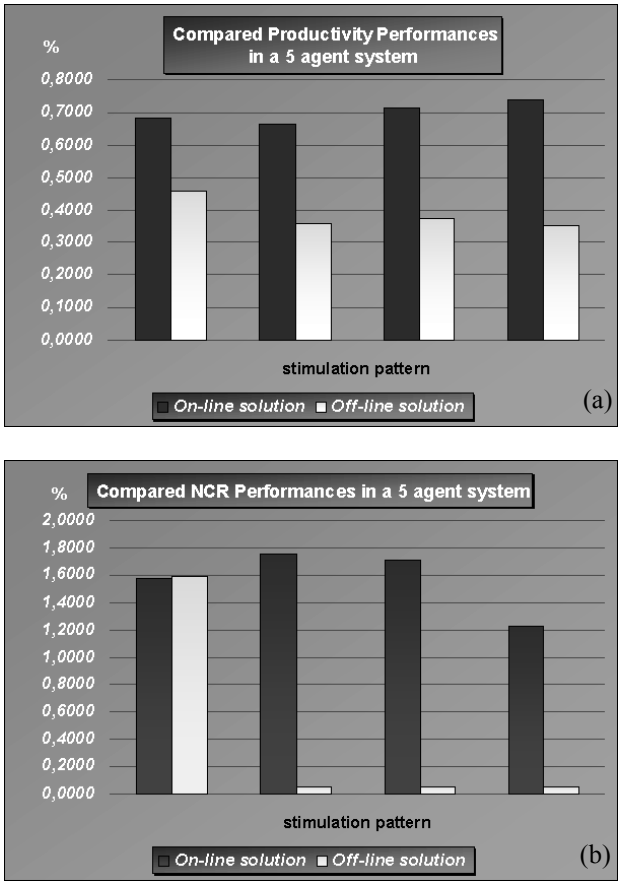

Fig. 6 (a,b): Diagrams of performances of run-time e off-line algorithms in a small center (5 agents).

The off-line algorithm has been sized fixing a NCR_0 value of 1%, that is with a maximum busy server

probability of NCR_0 / p. This step confirms the main theoretical limitation of this algorithm: Hit Rate may change strongly during campaigns (even if slowly) from expected values, changing the equivalent arrivals rate in the adopted Markov model.
Test results confirmed queue model limitations, outlined in previous sections, mainly predictable in small centers. A time analysis of the evolution of output metrics for the on-line solution evidenced its convergence and stability, in addition to its intrinsic flexibility and adaptivity which improve global performances.

5 Conclusions

This paper focuses on predictive dialling algorithm solutions in outbound dialling systems. Some possible models have been introduced, describing typical quantities in the telephony world and in particular in predictive dialling applications. Some numerical problems have also been outlined suggesting possible workarounds. Particular attention has been given to the choice of significant algorithm models for the kernel problem of predictive dialling systems: the launching engine. The commonly used Markov model has been analyzed, discussing theoretical limitations in its real world application.
It was also discussed how Chapman-Kolmogorov limit theorems become less significant in small centers, mainly when characteristic parameters change strongly and when homogeneity and stationary features of stochastic processes in the model are hardly applicable. This consideration can't be always set aside, and in some situations can't be balanced by the law of large numbers.
The proposed algorithm belongs to the on-line greedy category and is mainly based on a launching choice, that depending on system state iterates on it and looks for a statistical local optimum. Some corrections have been suggested to overcome intrinsic limitations of this model. Moreover, some ideas have been proposed for this algorithm model, outlining its flexibility. A new point of view has been approached leading to a different model that gives new possibilities and tools to improve algorithm tuning.

References:

[1] A. Evans, The future sound of CTI, *Voice International*, Vol. 4, No. 5, June 1997, p. 26.
[2] M. Cortese, *Sistemi I.V.R. e il Predictive Dialling*, Tesi dell'Università di Padova, 1998.
[3] A.Szlam and K. Thatcher, *Predictive dialling Fundamentals*, Flatiron Publishing Inc, 1996.
[4] G. Moscaletti and G.P. Almirante, *Call Center, come progettarlo, come realizzarlo, come gestirlo*, edipi, 1997.
[5] R. Nelson, Probability, *Stochastic processes, and Queueing Theory*, Springer-Verlag, 1995.
[6] J. Medhi, *Stochastic Models in Queueing Theory*, Academic Press, 1991.
[7] T.H. Cormen, C.E. Leiserson, R.L. Rivest, *Introduction to Algorithms*, MIT Press, 1990.
[8] A.V. Oppenheim e R.W. Shafer, *Discrete-Time Signal Processing*, Ed. Prentice Hall, 1989.