# Maximization of the Average Quality of Anytime Contract Algorithms over a Time Interval

Arnaud Delhay, Max Dauchet, Patrick Taillibert, Philippe Vanheeghe

# Maximization of the Average Quality of Anytime Contract Algorithms over a Time Interval

Arnaud Delhay*
LIFL - ISEN
F-59046 Lille
*aclel @ isen.fr*

Max Dauchet
LIFL
F-59655 Villeneuve d' Ascq
*Max. Dauchet @ lifl.fr*

Patrick Taillibert
Thomson-CSF Detexis
F-78190 Trappes
*Patrick. Taillibert @ detexis. thomson-csf com*

Philippe Vanheeghe
ISEN
F-59046 Lille
*pva @ isen.fr*

## Abstract

Previous studies considered quality optimization of anytime algorithms by taking into account the quality of the final result. The problem we are interested in is the maximization of the average quality of a contract algorithm over a time interval. We first informally illustrate and motivate this problem with few concrete situations. Then we prove that the problem is NP-hard, but quadratic if the time interval is large enough. Eventually we give empirical results.

## 1 Introduction

Hard problems like planning or decision making cannot be reasonably treated by complete methods. That is the reason why [Dean and Boddy, 1988] first considered anytime algorithms, also called flexible algorithms in (Horvitz, 1988]. These algorithms offer a trade-off between time and performance. They are characterized by a *Performance Profile* [Grass, 1996] that enables a prediction about the quality of the results given by the algorithm depending on the execution time duration. This method has been used to solve several problems in various domains like robot control [Zilberstein and Russel, 1993], knowledge-based computation [Mouaddib and Zilberstein, 1995] and reactive agents [Adelantado and de Givry, 1995].

Quality is not the essential characteristic of a computation result: what really matters is its utility. The intuitive idea is that, in many situations, the utili' of a result decreases over time, and a result of medium quality rapidly obtained is more useful than a result of high quality obtained after a long time [Zilberstein and Russel, 1996]. But, when the algorithm operates on an uncertain environment, utility can be of another nature. This is the case in the following examples, which are all associated to a "crisis situation":

- a person (P) has to give his boss (B) a report in the morning. P only knows that B will ask for the report at some time between 8 a.m. and 11 a.m. The problem for P is that trying to achieve the best quality would be a good strategy only if the report were claimed at 11. If it is not the case, no report at all is available! Hence it seems better to ensure a medium quality draft for 8 a.m. and, once the draft is ready, to start to write a better quality report, expecting that the claim will occur late in the morning.
- Every time the enemy is going to launch a satellite, only the temporal window on which the event will occur is known in advance. Tb perturb the launching, some electromagnetic jamming action must be set up (planes have to take-off, lures must be activated...). All these actions take time and the best jamming is useless if achieved after the launch. What could be considered is to set up the jamming in order to ensure the best *average* quality on the time interval, then to maximize the utility (in the long term).
- When a tornado is announced, very little time is available to prepare oneself (and one's house) for any possible destruction. Hence it is important to achieve the best "utility" of the protections set in place, e.g. to ensure that minimal actions have been taken first (securing the kids), before improving the quality of the protections (nailing down the shutters).

In the previously described situations, an interruptible algorithm would be the best solution: at the time of the event, the best possible quality would be achieved. Unfortunately, interruptible algorithms are not always available since:

- none of the available algorithms might be interruptible,
- anytime algorithms might result from the composition of elementary interruptible algorithms. In that case, the result is of the contract kind [Zilberstein and Russel, 1996],
- contract algorithms can be transformed into interruptible ones [Zilberstein and Russel, 1996] but, notwithstanding the fact that the execution time is 4 times longer, this method only applies if the contract durations can be chosen freely (exponential series) which is generally not the case.

Finally, even with a genuine interruptible algorithm, situations exist in which the contract case re-occurs: if applying the results of the algorithm causes a change in the environment, it is no longer possible to let the algorithm continue in order to improve the solution from the previously

* This author is supported by the *Delegation Gene rale pour i'Armement (DGA),* French Ministry of Defense

obtained one; it must be restarted from scratch. For instance, this might occur in counter-measure applications, if the result of the interruptible algorithm is a jamming action which itself provokes a counter-jamming decision from the enemy.

In [Delhay *et al,* 1998], we proposed partial solutions to solve these kinds of problems consisting of maximizing the average quality over a time interval. But the results achieved are limited to convex quality functions (whose second derivative is positive) which are the less probable ones in real applications.

In this paper, after restating the problem of maximizing the average quality of an anytime contract algorithm over a time interval (section 2), we present general results about any kind of quality function approximated by a stepwise function. We prove that the problem is NP-hard, but becomes quadratic if the time interval is large enough (section 3). Wc then present empirical results which augurs well for the practical applicability of the approach (section 4).

## 2 Maximizing the average quality over a time interval

We use the notion *of* contract algorithms which was first coined by Zilberstein [Zilberstein, 1993], even though they appeared before, like RTA* [Korf, 1985]. Contract algorithms can also result from the composition of anytime modules. In this paper, we assume that the performance profiles are deterministic functions of time. It is sometimes difficult to construct them with such a confidence, but they are good approximations of the performance profiles used in real situations.

### 2.1 Informal presentation

A typical example is the following: an attack might happen with a uniform probability over a given time interval, and a contract algorithm, whose performance profile is increasing over time (figure 1) is available to counter-attack. The problem then consists in determining how to best prepare the counter-attack in order to get the best chance of survival over the time interval.

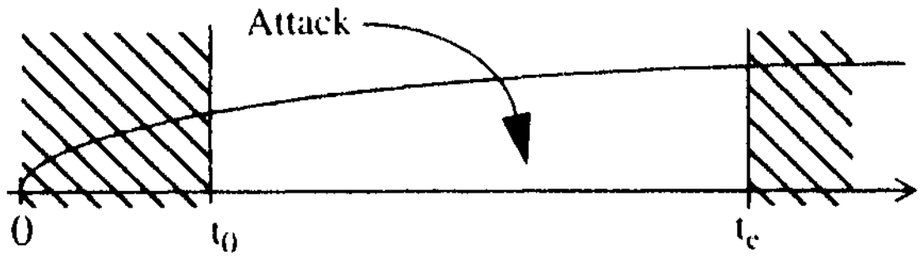


***figure 1: The attack might happen at any time over $[t_0, t_c]$***

An answer to the attack must be given between $t_0$ and $t_c$ and it is possible to begin the computation at time t = 0. To answer to this problem, several solutions can be considered.

First (figure 2) we activate the algorithm for a short contract ($t_1$); in that case, the result has a relatively poor quality but this quality is available on a relatively long time interval ($t_e$ - $t_1$).

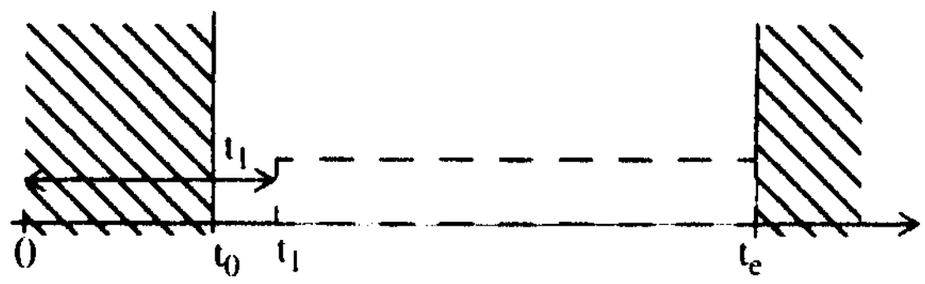


*figure 2: short but bad preparation*

To get a better quality, we have to start the algorithm with a longer contract ($t_2$): the quality of the result is better, but most of the time, on $[t_0, t_2)$, no "quality" (that is, no protection from the attack) is available (figure 3).

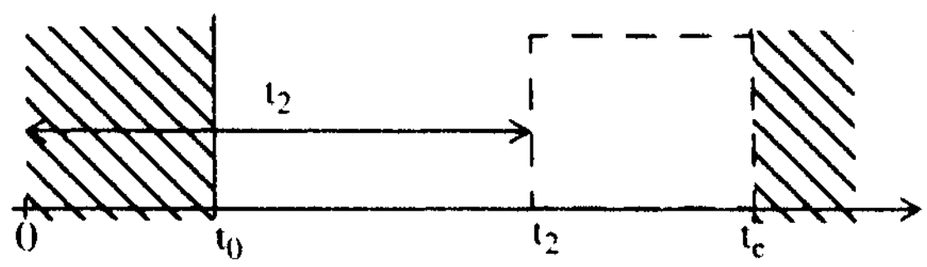


***figure 3: good but slow preparation***

These two cases lead to a simple observation: if the algorithm starts with a (sufficiently) short contract ($t_i$), then the remaining time can be used to restart the algorithm with a contract $t_2$ to get a better result.

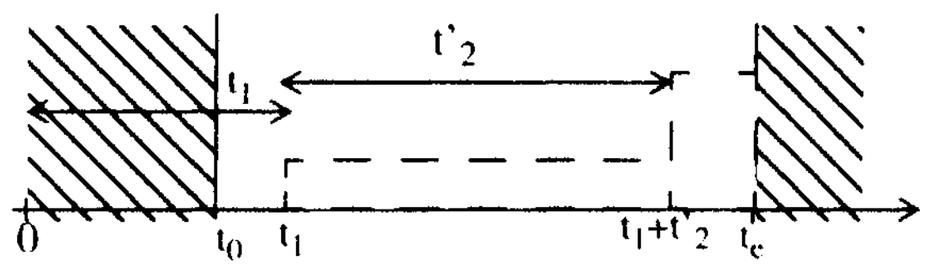


***figure 4: mixed preparation***

Hence an effective method is to start the contract algorithm several times to get a good cover of the time interval and a minimal quality early. We only have to respect two conditions:

- The sum of all contracts must be strictly lower than the length of the interval where it is possible to compute
- Every new contract must be longer than the previous one to improve the quality of the result

*Remark:*

Note that to maximize the average quality, we never execute more than one contract before $t_0$. Should we do that, all contracts before $t_0$, except the last, would be useless.

### 2.2 Formalization

Now let us give the definition of the average quality over an interval where $\langle\Theta_n\rangle = \{\theta_1, \theta_2, \ldots, \theta_n\}$ denotes the duration of successive runs of the algorithm (contracts).

definition 1 : *integral quality*

*Let f be the performance profile of a contract algorithm A. The* integral quality *of A over a time interval $[t_0, t_e]$, relative to a cl $\langle\Theta_n\rangle$ of n contracts with performance profile f is defined as follows:*

$$Q(f,\langle\Theta_n\rangle) = min(0,\theta_1 - t_0)\cdot f(\theta_1) + \sum_{i=1}^{n-1} f(\theta_i)\cdot\theta_{i+1} + f(\theta_n)\cdot\left(t_e - \sum_{i=1}^{n}\theta_i\right)$$

*The above definition is only usable if the contracts respect the sum constraint, that is:* $\sum_{i=1}^{n}\theta_i < t_e$

*The* average quality $\bar{Q}(f,\langle\Theta_n\rangle)$ *is equal to the integral quality divided by the length of the interval* $[t_0, t_e]$.
*Q(f) is the supremum of the integral quality, and Q(f) the supremum of the average quality.*

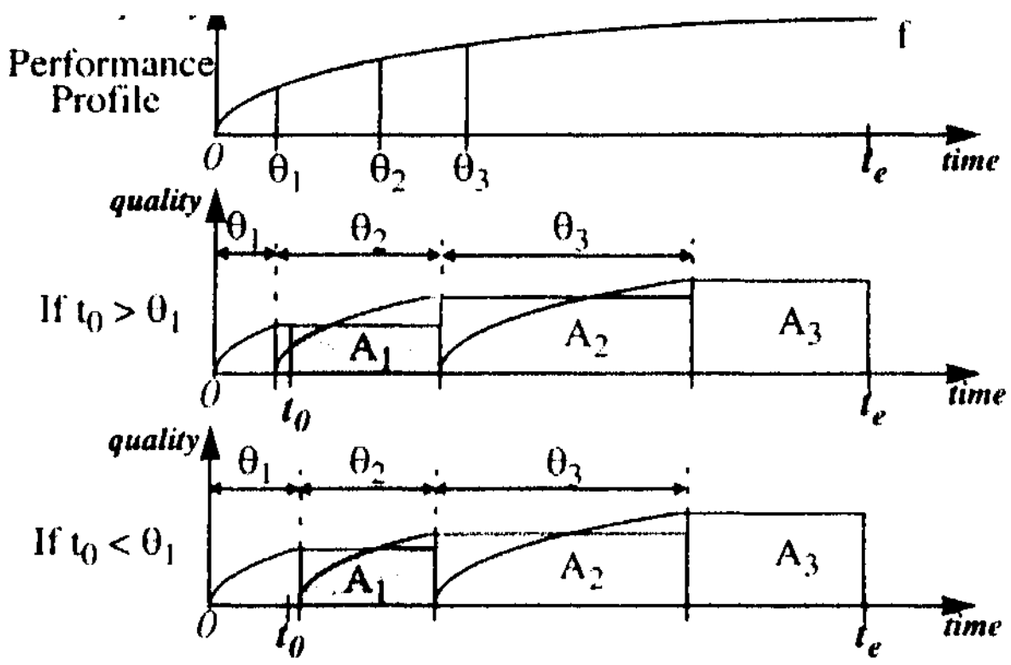


***figure 5: Performance Profile and Average Quality***

## 2.3 The analytic case

In [Delhay *et al.,* 1998], we studied the maximization of the average quality for continuous and derivable performance profiles, leading to preliminary analytic results that partially cover functions with constant curvature, that is the convex case and the concave case for one contract. These results are shown below.

A first theorem gives the value of a single contract. The linear performance profile is a limit and the single contract equals $t_e/2$. For convex perfor ance profiles, the single contract is greater than $t_e/2$. In the concave case, it is lower than $t_e/2$. A second theorem states that, for convex performance profiles, there is no need to start several contracts over $[0,t_e]$, as a single contract always gives the best average quality. The value of this contract $\theta_1$ is analytically defined by a simple equation.

$$\theta_1 = t_e - \frac{f(\theta_1)}{f'(\theta_1)}$$

The point we can add is that there is at least two contracts in the concave case because of the first theorem. As a matter of fact, as the single contract is lower than $t_e/2$., it is possible to add a contract greater than the first one, but respecting the sum constraint, that improves the average quality.

# 3 Average quality for stepwise constant performance profiles

Because of the difficulties involved in solving this problem in the continuous and derivable case, we chose to solve the problem by approximating the performance profile in order to get a discrete problem. We first tried using a stepwise linear function. Even with this approximation, we did not manage to exhibit interesting properties. Then we approximated the performance profiles with stepwise constant functions. This approach not only enables us to avoid difficulties due to the continuous and derivable performance profiles, but also makes sense in the common representation of performance profiles, that is the discrete tabular representation. Moreover, lemma 1 states that the average quality is approximated with the same error as the error on performance profile itself. In particular, this is true when approximating the continuous performance profile f with a stepwise function.

lemma 1 : *approximation lemma*

*If* $|f-g|<\varepsilon$ *then* $|\bar{Q}(f)-\bar{Q}(g)|<\varepsilon$

*Proof:* This is a property of integrals applied to definition 1.

So, hereafter the performance profiles are stepwise constant functions, either originally, or by approximation. A stepwise constant function is a function such as, for a finite set of thresholds $\{\theta_1,\ldots,\theta_n\}$, f is constant on every interval $[\theta_i, \theta_{i+1})$. The following lemma allows us to treat the maximization problem as a discrete problem, by only examining the steps of the performance profile instead of all values in the interval $[0,t_e]$.

lemma 2 : *(stepwise function lemma)*

*To maximize average quality, it is sufficient to choose all contracts* $\theta_i$ *in the set of thresholds of the stepwise function.*

*Proof:*

If a contract G is in the interval $(\theta_i,\theta_{i+1})$, replacing $\theta$ by $\theta_i$ increases the average quality. It can be checked by calculating the difference between the two average qualities (with and without $\theta$)

Thanks to lemma 2, we call this (discrete) problem MAXQSF (MAXimization of the average Quality for a Stepwise Function). $MAXQSF(f,t_e)$ denotes the maximum of the integral quality of an increasing stepwise function f over the time interval $[0,t_e]$. In the next subsections, the tractability of MAXQSF is considered.

## 3.1 MAXQSF is NP-hard

The following theorem states that the MAXQSF problem is intractable in general.

theorem 1 : *MAXQSF(f, $t_e$) is NP-hard*

The proof consists in reducing polynomially the Knapsack problem to MAXQSF. The Knapsack $(\langle A\rangle,b)$ problem is to find a part of $\langle A\rangle=\{a_1,a_2,\ldots,a_n\}$, a set of naturals, whose sum of its elements is maximal and lower than a natural b. We

assume that the aj are sorted and distinct. This restriction is still NP-complete. The reduction consists in building a stepwise constant function $f_{<A>}$ of 2n+1 steps that gives an instance of MAXQSF(f,$t_e$) for any instance of Knapsack(<A>,b).

To do so, we use the fact that the increase of average quality obtained by adding a contract $\theta'$ between two consecutive contracts $\theta$ and $\theta''$ only depends on $\theta$ and $\theta''$ and $\theta_\varphi$ (the last contract).

The variation of the average quality induced by the choice of introducing a threshold as a contract depends on the other choices and is not easy to evaluate. That is why $f_{<A>}$ is constructed by alternating the thresholds (odd numbers) that corresponds to the elements of <A>, and thresholds (even numbers) that are chosen to necessary belong to the optimal choice of the best average quality. The even thresholds "isolate" the effects of a choice in the set of odd thresholds; hence, the improvement of the average quality obtained by adding an odd threshold (element of <A>) only depends on the two even thresholds surrounding it[1]. That is the reason why we chose them so that if the corresponding element $\alpha$ of <A> is in the solution, the improvement of the average quality is $\alpha$. As a consequence, maximizing the average quality will maximize the load of the knapsack.

The reduction also requires that optimizing the average quality satisfies the sum constraint of the Knapsack problem. This is obtained by choosing $t_c$, the end of the time interval, such that:

$$t_e = b + \sum_{i \text{ even}} s_i + 0.5$$

Since the sum of all contracts in MAXQSF must be less than $t_e$ and that all the even thresholds belong to the solution by construction, we therefore have:

$$\sum_{i \text{ odd}} s_i \leq b$$

which is the sum constraint of the Knapsack problem.

We do not have enough space to include all the steps necessary to perform the reduction we have presented. Let's just say that the choice *of* the f function is done such that $f_i - f_{i-1} = K^{4n-i}$ with K great enough (but polynomially linked to the size of <A>) and that the $s_{2i}$ are chosen such that $s_{2i} < a_i + 1/(3nK^{2n})$. This construction is polynomial in the size of <A> which proves that MAXQSF is NP-hard. For a detailed proof, see [Delhay and Dauchet, 1999].

Theorem 1 looks like an instance of theorem 4.2 presented by [Zilberstein and Russel, 1996] in the framework of composition of anytime algorithms. However, it is not the case: we look for the best average quality, whereas they look for the best final quality after a fixed contract, the most important difference being that we do not know the number of contracts necessary to get the best average quality.

*J .also on the last conti($\theta_\varphi$)., but, since it's an even one, it belongs to the optimal choice.*

## 3.2 MAXQSF with no sum constraint is quadratic

Our problem is intractable in general. The main difficulty comes from the search over the set of thresholds assuming that the sum constraint does not allow to take any subset of thresholds. As in Knapsack, it is necessary to judiciously choose the candidate contracts. So we could suppose that:

$\sum_{i=1}^{p} s_i < t_e$ where $S_i$ is a threshold of the performance profile f.

Hence it is possible to add any contract because the time interval is large enough to contain all contracts. That restricts MAXQSF and leads to a lower complexity algorithm.

The idea of this dynamic programming algorithm is founded on lemma 3 which allows the division of the set of combinations of thresholds into distinct subsets composed of the combinations finishing by a fixed threshold $s_i$. The lemma allows to iteratively compute $Q(f,<\Theta|\{s_i\}>)$,, noted MAXQ(s,) for subset of combinations finishing by $s_i$ with $S_i$ from $S_i$ to $s_p$, to finally give $MAXQ(s_p) = Q(f,<\Theta|\{s_i\}>)$ is the ordered set $\Theta$ finishing by $s_i$

lemma 3 :

*Let $S_i$ and $S_j$ be two thresholds of<s>, the set of thresholds off, with j<i.*
*For any* $s_i$, *let MAXQ(Sj) be the maximum of integral quality with .*$S_i$, *as the last contract an*$TR(s_i)=(t_e-\theta_1-\ldots-s_i)$ *the remaining time for the optimal choice of contracts with Sj as the last contract.*
*Then* $MAXQ(s_i) = max_j (MAXQ(s_j) + (TR(s_j) - s_i) \cdot (f(s_i) - f(s_j)))$

*and especially: MAXQ(s ) =* Q(f), *where $s_p$ is the last threshold of <s>.*

*Proof:*

- It can be immediately proved that $Q(f,<\Theta|\{s_j,s_i\}>) = Q(f,<\Theta|\{s_j\}>)+(TR(s_j)-s_i).(f(s_i)-f(s_j))$ By using the maximum for both members, the result of the lemma comes as TR(S:) only depends on s;. Indeed, $TR(s_j)$ is the remaining time for the optimal choice of contracts with $s_j$, as the last contract.
- For the second part of the lemma, note that $s_p$ is necessary in the optimal choice of contracts. It could always be added to the choice of contracts (as there is no sum constraint) and improves the quality. By definition, MAXQ(Sp) gives the maximal integral quality at the last step of iteration.

The complexity of the algorithm is $O(n^2)$, with n the number of thresholds.

*Algorithm:*

Only **one threshold** $\theta_1 \leftarrow s_1$ and $TR(s_1) \leftarrow t_e - max(t_0, s_1)$
$MAXQ(s_1) \leftarrow TR(s_1) \cdot f(s_1)$
CHOICE($S_J$) denotes the optimal choice of contracts with $S_i$ as the last contract.

*For* i *from* 2 *to* p *Do*
  $Q_{temp} \leftarrow 0$
  $TR_{temp} \leftarrow 0$
  $CHOICE_{temp} \leftarrow \varnothing$
  *For* j *from* 1 *to* i-1 *Do*
    *If* $MAXQ(sj) + (TR(s_j) - s_i) \cdot (f(s_i) - f(s_j)) > Q_{temp}$
    *Then*
    $Q_{temp} \leftarrow MAXQ(s_j) + (TR(s_j) - s_i) \cdot (f(s_i) - f(s_j))$
    $TR_{temp} \leftarrow TR(s_j) - s_i$
    $CHOICE_{temp} \leftarrow CHOICE(s_j) \cup \{s_i\}$
    *Endif*
  *Endfor*
  $MAXQ(s_i) \leftarrow Q_{temp}$
  $TR(s_i) \leftarrow TR_{temp}$
  $CHOICE(s_i) \leftarrow CHOICE_{temp}$
*Endfor*
$Q(f) \leftarrow MAXQ(s_i)$
$CHOICEMAX \leftarrow CHOICE(s_i)$

## 4 Empirical results

The results obtained so far do not take into account the duration of the deliberation itself. It might be a serious impediment to the practical application of our approach since we proved that the general problem is NP-hard! Fortunately, achieving the optimal solution is not a necessary condition for applying our method: a set of contracts approaching the optimal value of the average quality on the time interval is sufficient. That is the reason why we conducted a set of experiments designed to estimate the practical complexity of our problem. And the result was far better than expected: for all the cases we studied, the average quality achieved by the best choice *oi* a set of 2 contracts was always at a distance lower than 2.75% from the quality of the optimal choice.

These results concern a family of monotonic functions which approximate the quality functions most often encountered in practical cases. That is the reason why we think that these results can be of some general interest.

The experiment was twofold. First, we studied the contribution of the shape of the quality function to the value of the average quality. For that, we considered the family of functions defined by the following equation where the parameter "a" permits the control of the curvature of the function as shown in the associated graphics.

$$f(t) = \frac{a \cdot t}{(a-1) \cdot t + 1}$$

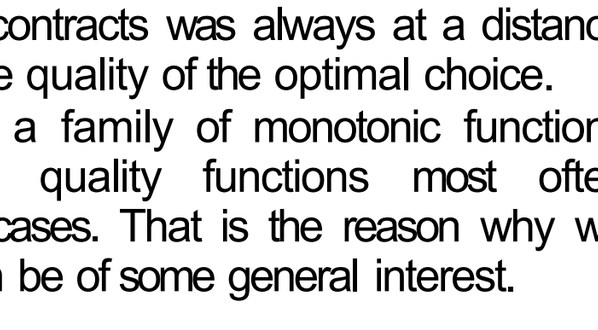


*figure 6: performance profile for experiments*

Every function ("a" varying from 1 to 205 incremented by 5) was approximated by a stepwise function of 50 steps for which we computed the optimal average quality; we also computed the error w.r.t. this optimal value when considering smaller sets of contracts. We relied upon integer programming in order to avoid the classic problems of floating point numbers and took advantage of the sum constraint to limit the computation time. The results are summarized in the following table where a is the curvature, n is the number of contracts and * is the optimal solution:

| a \ n | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 0.1 | * | | | | | |
| 1 | * | | | | | |
| 5 | 10.83% | 0.23% | * | | | |
| 25 | 12.33% | 2.47% | 0.25% | * | | |
| 45 | 11.59% | 2.71% | 0.59% | * | | |
| 65 | 10.75% | 2.63% | 0.6% | 0.01% | * | |
| 85 | 9.97% | 2.44% | 0.53% | * | | |
| 105 | 9.52% | 2.36% | 0.46% | 0.01% | * | |
| 125 | 8.98% | 2.29% | 0.45% | 0.02% | * | |
| 145 | 8.55% | 2.22% | 0.39% | 0.02% | * | |
| 165 | 8.29% | 2.02% | 0.35% | 0.01% | * | |
| 185 | 7.96% | 1.83% | 0.28% | * | | |
| 205 | 7.54% | 1.68% | 0.31% | * | | |

*table 1: Error for limited number of contracts*

For $a \leq 1$ the optimal quality is obtained with only one contract which was the expected result because of theorem 2. In the other cases, it is clear that the optimal quality obtained with 2 contracts is very close to the best result obtained notwithstanding the number of contracts.

We also investigated the influence of the temporal location of the time interval $(t_0)$, which was set to 0 in the first experiment, while $t_c$-$t_0$ remained equal to 1. This might be of importance for many applications when time is available before the "attack" might occur. Here again, as shown on the following figure, restricting the deliberation to the computation of only 2 contracts gives very accurate results (even if only one contract rapidly gives very good results):

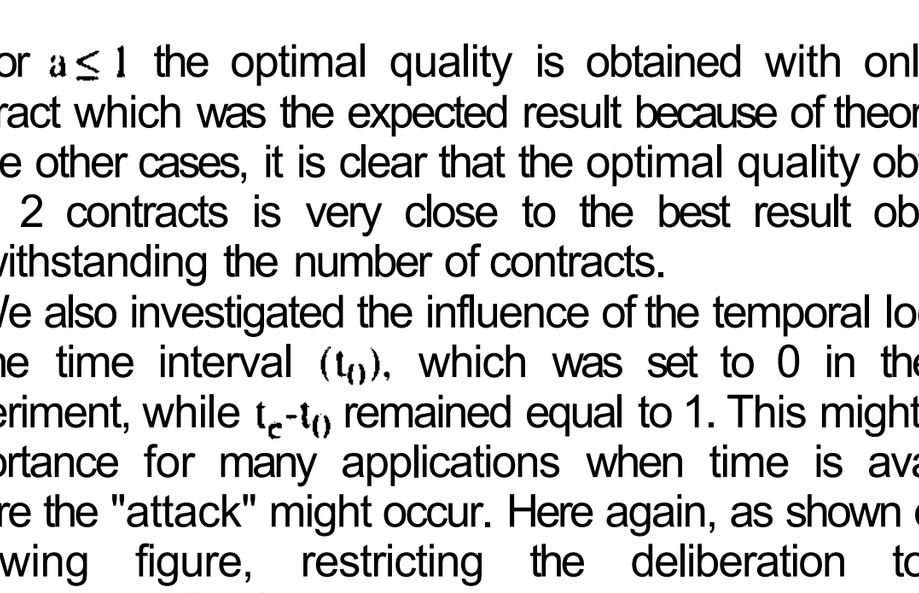


*figure 7: Influence of $t_0$ on the number of contracts*

In that experiment, the curvature was set to 51 and, since $t_e$ was increased, the "sum constraint heuristic" became less efficient; hence we operated with a stepwise function of only 30 steps to keep the computation of optima tractable.

# 5 Conclusion

The aim of this paper was to present an off-line optimization of the time allocation for a contract algorithm so as to get the best chance of survival over a time interval. So far, no analytical method for solving the problem of maximizing the average quality over a time interval is known in the general case. That is the reason why we proposed solutions for discrete performance profiles. This restriction is not a real impediment since:

- a discrete tabular representation for the performance profiles is not a rare occurrence,
- the average quality resulting from a discrete performance profile can be as close as necessary to a given continuous performance profile (lemma 1).

We showed that the so-called MAXQSF problem is NP-hard in general, and quadratic if the time interval is large enough. This, unfortunately, is not frequently encountered in practical applications. Nonetheless, the experiments we carried out lead us to think that the "practical" complexity of that problem is quite low, which makes it possible to use the approach in real-lime applications. Further studies to find a theoretical explanation of this behavior could prove very interesting.

The average quality of a contract algorithm A can be defined by:

$$\frac{1}{t_e - t_0} \cdot \int_{t_0}^{t_e} f^*(t)dt$$

where f is the quality of an intcrruptiblc algorithm A associated to A by the relation: $A^*(t) = A(\theta_i)$ where $\theta_i$ is the last contract executed at t.

[Zilberstein and Russel, 1996] gives a simple construction to transform a contract algorithm A of quality f into an interruptible algorithm A of quality f such as $f^*(4t) > f(t)$ The first difference with our study is that we consider the case of a given time interval to optimize the average quality. In our case, this leads to a construction of A which optimizes the average quality. The second difference is that our problem permits cases where the length of each contract is imposed by a method, when Zilberstein and Russel assume that the length of each contract can be arbitrarily chosen.

Note that a uniform probability of appearance of the attack over the interval has been considered. There arc situations where this probability is not constant, such as with a gaussian. A future study could concentrate on this point.

Another extension could concentrate on our evaluation criteria, that is the average quality. Even if this criteria gives the best statistical results, there could exist others, for example that could take the number of contracts into account (a good solution therefore is a choice of few contracts).

# Acknowledgments

I am especially appreciative of the discussions I had with Dominique Lohez at different stages of this work. His remarks and suggestions were very valuable.

# References

[Adelantado and De Givry, 1995] M. Adelantado, S. De Givry, *Reactive/anytime Agents, Towards Intelligent Agents with Real-Time Performance,* IJCAI'95, Workshop on Anytime Algorithms and Deliberation Scheduling, August 21-25, 1995, Montreal, Canada.

[Dean and Boddy, 1988) Thomas Dean and Mark Boddy, *An Analysis of Time-Dependent Planning,* AAAI 88, August 1988, Saint Paul Minnesota.

[Delhay *el al,* 1998] Arnaud Delhay, Max Dauchet, Patrick Taillibert, Philippe Vanheeghe, *Optimization of the Average Quality of Anytime Algorithms,* Workshop on Monitoring and control of real-time intelligent systems, ECAF98, August 1998, Brighton, pp. 1-4.

[Delhay and Dauchet, 1999] Arnaud Delhay and Max Dauchet, *Complexity of the maximization of the average quality of anytime contract algorithms,* Internal Report L1FL-99-09, Laboratoire dTnformatique Fondamentale de Lille, 1999, ftp://ftp.lifl.fr/pub/reports/intemal/1999-()9.ps.gz

[Grass, 1996] Joshua Grass, *Reasoning about computational resource allocation, An introduction to anytime algorithms,* Crossroads, ACM student magazine, University of Massachusetts, September 1996, http:// anytime.cs.umass.edu/~jgrass/school/papers/racra.hlml.

[Horvitz, 1988] Fric J. Morvitz, *Reasoning under varying and uncertain resource constraints,* Proceedings of the Seventh National Conference on Artificial Intelligence, Minneapolis, MN. August 1988, Morgan Kaufmann, San Mateo, CA. pp. 111-116.

[Korf, 1985] Richard E. Korf, *Real-Time Heuristic Search,* Artificial Intelligence vol. 42, 1985, pp. 189-211.

[Mouaddib and Zilberstein, 1995] Abdel-Illah Mouaddib, Shlomo Zilberstein, *Knowledge-Based Anytime Computation,* in Proceedings of the 14th 1JCAI, Montreal, Canada, 1995, pp 775-781.

[Zilberstein and Russel, 1993] Shlomo Zilberstein, Stuart J. Russel, *Anytime Sensing, Planning and Action: A Practical Model for Robot Control,* in Proceedings of the 13th 1JCA1, Chambery, France, 1993.

[Zilberstein and Russel, 1996) Shlomo Zilberstein, Stuart J. Russel, *Optimal composition of real-time systems,* Artificial Intelligence vol. 82, 1996, pp. 181-213.