Journal of Numerical Mathematics and Stochastics, **10** (1) : 58-72, 2018 © ***JNM@S***
http://www.jnmas.org/jnmas10-5.pdf Euclidean Press, LLC
Online: ISSN 2151-2302

Optimal Algorithm Re-Initialization for Combinatorial Optimization

G. SEBASTIANI, and D. PALMIGIANI

Istituto per le Applicazioni del Calcolo "Mauro Picone", CNR, Rome, and Istituto "Guido Castelnuovo", "Sapienza" Università di Roma, Italy, E-mail: sebastia@mat.uniroma1.it

Abstract. *We propose a new iterative procedure to find the best time for re-initialization of meta-heuristic algorithms to solve combinatorial optimization problems. The sequence of algorithm executions with different random initializations evolves at each iteration by either adding new independent executions or extending all existing ones up to the current maximum execution time. This is done on the basis of a criterion that uses a surrogate of the algorithm failure probability, where the optimal solution is replaced by the best so far one. Therefore, the new procedure can be applied in practice. We prove that, with probability one, the maximum time of current executions of the proposed procedure approaches, as the number of iterations diverges, the optimal value minimizing the expected time to find the solution. We apply the new procedure to several Traveling Salesman Problem instances with hundreds or thousands of cities, whose solution is known, and to some instances of a pseudo-Boolean problem. As base algorithm, we use different versions of an Ant Colony Optimization algorithm or a Genetic Algorithm. We compare the results from the proposed procedure with those from the base algorithm. This comparison shows that the failure probability estimated values of the new procedure are several orders of magnitude lower than those of the base algorithm for equal computational cost.*

Key words : Optimization Methods, Re-Initialization, Probability, Stochastic Processes.

AMS Subject Classifications : 60C99, 68R05, 65Y10

1. Introduction

Solving a combinatorial optimization problem (COP) consists of finding an element, within a finite search domain, which minimizes a given so called *fitness function*. The domain has typically a combinatorial nature, e.g. the space of the Hamiltonian paths on a complete graph. The COP prototype is the Traveling Salesman Problem (TSP), whose solution is a Hamiltonian

cycle on a weighted graph with minimal total weight [1]. Although a solution of a COP always exists, finding it may involve a very high computational cost. The study of the computational cost of numerical algorithms started in the early 1940s with the first introduction of computers. Two different kinds of algorithms can be used to solve a COP problem: exact or heuristic. A method of the former type consists of a sequence of non-ambiguous and computable operations producing a COP solution in a finite time. Unfortunately, it is often not possible to use exact algorithms. This is the case for instances of an *NP*-complete COP. In fact, to establish with certainty if any element of the search space is a solution, requires non-polynomial computational cost. Alternatively, heuristic algorithms can be applied. Such type of algorithms only guarantee either a solution in an infinite time or a suboptimal solution. Of great importance are the *meta-heuristic* algorithms (MHA) [2]. They are independent of the particular COP considered, and often stochastic. Among them, there are Simulated Annealing [13], Tabu Search [8], Genetic Algorithms (GA) [9] and Ant Colony Optimization (ACO) [4].

A natural issue for MHA concerns their convergence [7], [17], [10], [15], [11]. Due to the stochastic nature of such algorithms, they have to be studied probabilistically; unfortunately, even when their convergence is theoretically guaranteed, it is often too slow to successfully use them in practice. One possible way to cope with this problem is the so called *restart* approach, which, aside from the present context, it is used more generally for simulating rare events [5], [6], [14]. It consists of several independent executions of a given MHA: the executions are randomly initialized and the best solution, among those produced, is chosen. When implementing the restart on a non-parallel machine, the restart consists of periodic re-initializations of the underlying MHA, the period T being called *restart time*.

Despite the fact that the restart approach is widely used, very little work has been done to study it theoretically for combinatorial optimization [12], [16]. In [12], the restart is studied in its dynamic form instead of the static one considered here. Some results are provided for a specific evolutionary algorithm, i.e. the so called (1+1)EA, used to minimize three pseuso-Boolean functions. In [16] the fixed restart strategy is considered as done here. The first two moments of the random time T_R for the restart to find a solution (optimization time) are studied as a function of T. An equation for T is derived, whose solution minimizes the expected value of T_R, i.e. $\mathsf{E}[T_R]$. However, this equation involves the distribution of the optimization time of the underlying MHA, which is unknown.

In practice, the underlying MHA is very commonly restarted when there are negligible differences in the fitness of the best-so-far solutions at consecutive iterations during a certain time interval. This criterion may not be adequate when we want to really find the COP solution and we are not satisfied with suboptimal ones.

Here we propose a new iterative procedure to optimize the restart. Each iteration of the procedure consists of either adding new MHA executions or extending along time the existing ones. Along the iterations, the procedure uses an estimate of the MHA failure probability where the optimal solution is replaced by the *best so far* one. We recall that the failure probability of a stochastic algorithm $p(k)$ is the probability that the optimal solution has not been found up to iteration k.

We prove that, with probability one, the restart time of the proposed procedure approaches, as the number of iterations diverges, the value that minimizes $\mathsf{E}[T_R]$.

We also show the results of the application of the proposed restart procedure to several TSP instances, whose solution is known, with hundreds or thousands of cities. As MHA we use

different versions of the ACO algorithm Max-Min Ant System (MMAS) [18]. Based on a large number of experiments, we compare the results from the restart procedure with those from the MMAS. This is done by considering the failure probability of the two approaches for the same total computation cost. This comparison shows a significant gain when applying the proposed restart procedure. Similar results are obtained when applying the RP to the MMAS or to a GA for solving some instances of a pseudo-Boolean problem. The algorithms have been implemented in MATLAB or in C.

2. The Procedure

In this section, we first give some mathematical details of the restart, that will be then used to define the RP. In the following part, we present the procedure in details, also providing a pseudocode.

Following [3], the failure probability at iteration k of the restart is the probability that the optimization time of the restart T_R is larger than k:

$$\mathsf{P}(T_R > k) = p(T)^{\left\lfloor \frac{k-1}{T} \right\rfloor} p\left(k - \left\lfloor \frac{k-1}{T} \right\rfloor T\right), \tag{1}$$

where $p(\cdot)$ is the failure probability of the underlying MHA. We notice that the first factor in (1) is the probability that the optimal solution has not been found in each of the previous $\left\lfloor \frac{k-1}{T} \right\rfloor$ re-initializations. The second factor is the probability that the optimal solution has not been found up to iteration $k - \lfloor \frac{k-1}{T} \rfloor T$ of the current re-initialization. The restart failure probability is geometrically decreasing towards zero with the number $\left\lfloor \frac{k-1}{T} \right\rfloor$ of re-initializations, the base of such geometric sequence being $p(T)$. Therefore, a short restart time T may result in a high value of $p(t)$ and a slow convergence. On the contrary, if the restart time T is high, we may end up with a low number of re-initializations $\left\lfloor \frac{k-1}{T} \right\rfloor$ and a high value of the restart failure probability. Then, a natural problem is to find an "optimal value" of T when using a finite amount of computation time.

The restart could be optimized by choosing a value for T that minimizes the expected value of the time T_R:

$$\mathsf{E}[T_R] = \sum_{k=1}^{\infty} \mathsf{P}(T_R > k). \tag{2}$$

In fact for any random variable not negative X it is possible to write

$$\mathsf{E}[X] = \int_0^{\infty} \mathsf{P}(X > t)\, dt. \tag{3}$$

In our case, the random variable T_R is discrete and the integral in (3) is replaced by a series whose generic term is $\mathsf{P}(T_R > t)$.

We now derive an upper bound for the r.h.s. of (2):

$$\begin{aligned} \mathsf{E}[T_R] &\le \sum_{k=1}^{\infty} p(T)^{\left\lfloor \frac{k-1}{T} \right\rfloor} \le \sum_{k=1}^{\infty} p(T)^{\frac{k-1}{T}-1} \\ &= \frac{1}{(1-p(T)^{\frac{1}{T}})p(T)}. \end{aligned} \tag{4}$$

By means of this bound, we can then optimize the RP by minimizing the function $g(x) := \left[(1 - p(x)^{\frac{1}{x}}) p(x) \right]^{-1}$.

Whenever this function does not have a global minimum, there is no advantage to use the restart. In the other case, an optimal value for the restart time is provided by the first value t_m where the function g assumes its absolute minimum. However, this criterion cannot be applied in practice since the MHA failure probability is unknown.

The restart procedure (RP) starts by executing r_0 independent replications of the underlying MHA for a certain number of time steps T_0. Let us denote by $X_i(t)$ the solution produced by the replication i of the underlying algorithm at time t. Let $Y_i(t)$ be the fitness function value of the best solution found by the i-th replication until time t i.e. $Y_i(t) = \min\{f(X_i(s)), s = 1, \dots, t\}$, where f is the function to minimize. Each $\{Y_i(t), t = 1, 2, \dots\}$, $i = 1, 2, \dots$ is an independent realization of the same process. Then, at the end of iteration k, based on the criterion described later in this section, the RP either increases the number of replications from r_k to r_{k+1} by executing $r_{k+1} - r_k$ replications of the underlying algorithm until time T_k, or it continues the execution of the existing r_k replications until time $T_{k+1} > T_k$. Therefore, the RP can be described by a sequence $\{Y_{A_k}, k = 0, 1, \dots\}$ of nested finite matrices, extracted from the infinite matrix $\mathbf{Y}$:

$$\mathbf{Y} = \begin{pmatrix} \mathbf{Y_1(1)} & \mathbf{Y_1(2)} & \cdots & \mathbf{Y_1(t)} & \cdots \\ \mathbf{Y_2(1)} & \cdots & \cdots & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{Y_i(1)} & \cdots & \cdots & \mathbf{Y_i(t)} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix},$$

where $A_k := \{(i,t) : i = 1, \dots, r_k \quad t = 1, \dots, T_k\}$. The matrix Y_{A_k} corresponds to the first r_k rows and T_k columns of $\mathbf{Y}$. Let $\tilde{Y}_k$ denote the minimum value of the matrix Y_{A_k} at the end of iteration k: $\tilde{Y}_k = \min Y_{A_k} = \min_{(i,t) \in A_k} Y_i(t)$. We estimate the failure probability sequence by means of the empirical frequency

$$\hat{p}_k(t) = \begin{cases} \frac{1}{r_k} \sum_{i=1}^{r_k} 1_{\{Y_i(t) > \tilde{Y}_k\}} & t = 1, \dots, T_k, \\ 0 & \text{otherwise.} \end{cases}$$

Next, consider the function $g_k(t) = [(1 - \hat{p}_k(t)^{\frac{1}{t}}) \hat{p}_k(t)]^{-1}$, $t = 1, \dots, T_k$, and define $\hat{\sigma}_k$ the first time with a left and right increase of the function g_k (local minimum). Let λ be a number in $(0, 1)$. If $\hat{\sigma}_k < \lambda \cdot T_k$, then the RP increases the number of replications by means of a certain rule $r_{k+1} := f_r(r_k)$. Otherwise, the RP increases the restart time according to $T_{k+1} := f_T(T_k)$. We assume that $\forall x$ we have $f_r(x) > x$ and $f_T(x) > x$. As a consequence, for any fixed $x > 0$, it holds $f_r^{(k)}(x)$, $f_T^{(k)}(x) \to \infty$, k denoting the consecutive application of a function for k times. Therefore, the recursive formula for (r_k, T_k) is

$$(r_{k+1}, T_{k+1}) = \begin{cases} (f_r(r_k), T_k) & \text{if } \hat{\sigma}_k < \lambda \cdot T_k, \\ (r_k, f_T(T_k)) & \text{otherwise.} \end{cases}$$

Below there is the pseudocode for RP.

The RP pseudocode

$r = r_0$;
$T = T_0$;
for replication $i = 1, 2, \ldots, r$ **do**
perform execution A_i of MHA until time T_0;
save $\mathsf{A}_i(T_0)$;
end for
save Y_{A_0};
compute $\hat{\sigma}_0$ from Y_{A_0};
for iteration $k = 1, 2, \ldots$ **do**
if $\hat{\sigma}_{k-1} \geq \lambda \cdot T_{k-1}$ **then**
$T_k = f_T(T_{k-1})$;
$r_k = r_{k-1}$;
for replication $i = 1, 2, \ldots, r_k$ **do**
continue the execution of A_i until T_k;
save $\mathsf{A}_i(T_k)$;
end for
else then
$r_k = f_r(r_{k-1})$;
$T_k = T_{k-1}$;
for replication $i = r_{k-1} + 1, r_{k-1} + 2, \ldots, r_k$ **do**
execute A_i until T_k;
save $\mathsf{A}_i(T_k)$;
end for
end if
save Y_{A_k};
compute $\hat{\sigma}_k$ from Y_{A_k};
end for

---

3. RP Convergence

In this section, we describe some theoretical properties of the RP. The main result, i.e. theorem 3.2, concerns with the RP convergence. Specifically, we prove that, with probability one, the restart time of the RP approaches, as the iteration number diverges, the value t_m that minimizes $\mathsf{E}[T_R]$. This is done as follows. In lemma 3.1, we prove that the number of replications r_k diverges, and that the RP eventually finds the optimal solution. This lemma is then used to prove the technical lemma 3.2. Finally, this lemma and theorem 3.1 are used to prove the main result.

We denote by f_m the value of the solution of the optimization problem. Moreover, we recall the functions $g(t) = [(1 - p(t)^{\frac{1}{t}})p(t)]^{-1}$, where $p(t)$ is the failure probability and

$g_k(t) = [(1 - \hat{p}_k(t)^{\frac{1}{t}})\hat{p}_k(t)]^{-1}$, whose domain is $\{1, \dots, T_k\}$.
In order to derive the following results, we assume that

1. $g(\cdot)$ admits absolute minimum at the time t_m of its first local one, and it is strictly decreasing for $t \leq t_m$,
2. $p(1) < 1$.

We notice that if the absolute minimum of $g(\cdot)$ is reached after the time t_m of its first local one, the RP restart time still converges to t_m, so providing only a suboptimal value for the restart time. Point 2 in practice does not give limitations. In fact, we can always aggregate the first iteration of the algorithm with failure probability less than one and those before it into a single one.

Remark 3.1. We notice that, by the assumptions on the functions f_r and f_T, and by the RP definition, the probability that both the sequences r_k and T_k are bounded is zero.

Lemma 3.1. *Let $p(t)$ be as above. Let (r_k, T_k) be the sequence of random variables which describes the RP. Then,*
1. $P(r_k \to \infty) = 1$,
2. $P(\{\exists k : \quad \tilde{Y}_k = f_m\}) = 1$,

should hold.

Proof. **1**. If $\mathsf{P}(r_k \to \infty) < 1$, then, with positive probability, the following three conditions hold for a certain positive integer r:
i) $r_k = r$ eventually;
ii) T_k diverges (for remark 3.1);
iii) $\hat{\sigma}_k \geq \lambda T_k$ eventually (from ii) and the definition of the RP).
Eventually, there are only two mutually exclusive possibilities: either the underlying r copies of the algorithm have all reached the minimum or only some $r' < r$ of them will have experienced it. In both cases, it follows that, $\hat{p}_h(t)$ as well as $g_h(t)$ will not change for h large enough. Hence, eventually $\hat{\sigma}_h$ does not change as h increases, which is a contradiction with iii). Therefore $\mathsf{P}(r_k \to \infty) = 1$.

2. By i), since $p(1) < 1$, with probability one, there exists i such that $Y_i(1) = f_m$; for all k so large that $r_k \geq i$ it will be $\tilde{Y}_k = f_m$. This proves the point. ■

Lemma 3.2. *For each $t \in N$,*

$$P\left(\left\{\sup_k T_k < t\right\} \cup \left\{\sup_k T_k \geq t, \ \lim_{k\to\infty} \hat{p}_k(t) = p(t)\right\}\right) = 1,$$

holds .

Proof. Let us consider the case when the event $E_t := \left\{\sup_k T_k \geq t\right\}$ happens, then we can eventually compute $\hat{p}_k(s)$, for $s = 1, \dots, t$. By point 1 of lemma 3.1 and the strong law of large numbers, we get

$$\mathsf{P}\left(E_k,\ \lim_{k\to\infty} \tfrac{1}{r_k} \sum_{i=1}^{r_k} 1_{\{Y_i(t) > f_m\}} = p(t) \right) = \mathsf{P}(E_k).$$

Hence, using point 2 of lemma 3.1, we obtain that

$$\mathsf{P}\left(E_k,\ \lim_{k\to\infty} \tfrac{1}{r_k} \sum_{i=1}^{r_k} 1_{\{Y_i(t) > f_m\}} = p(t) \right)$$

is equal to

$$\mathsf{P}\left(E_k,\ \lim_{k\to\infty} \tfrac{1}{r_k} \sum_{i=1}^{r_k} 1_{\{Y_i(t) > \tilde{Y}_k\}} = p(t) \right)$$

Since, by definition

$$\hat{p}_k(t) = \tfrac{1}{r_k} \sum_{i=1}^{r_k} 1_{\{Y_i(t) > \tilde{Y}_k\}},$$

we have

$$\mathsf{P}\left(E_k,\ \lim_{k\to\infty} \hat{p}_k(t) = p(t) \right) = \mathsf{P}(E_k).$$

from which the thesis follows. ■

Theorem 3.1. *For the RP it holds that*

$$\mathsf{P}\left(\sup_k T_k > \frac{t_m}{\lambda} \right) = 1.$$

Proof. Let us assume that the thesis is not true. Then, there exists an integer number M such that $M \leq \frac{t_m}{\lambda}$ and $\mathsf{P}\left(\left\{ \sup_k T_k = M \right\} \right) > 0$. On the event $\{\sup_k T_k = M\}$, by both lemma 3.2 and the continuous mapping, we have the convergence $g_k(t) \to g(t)$, for any $t \leq M$. This means that, for any $\varepsilon > 0$ there is a positive probability that $\bigcap_{t=1}^{M} \{|g_k(t) - g(t)| < \varepsilon\}$, when k is large enough. Let us define $\tilde{M} := \min(M, t_m)$. We then have $g_k(\tilde{M}) < g(\tilde{M}) + \varepsilon$ and $g_k(t) > g(t) - \varepsilon$ for any $1 \leq t < \tilde{M}$. Subtracting the first inequality from the last one, we obtain

$$g_k(t) - g_k(\tilde{M}) > g(t) - g(\tilde{M}) - 2\varepsilon.$$

Since g is strictly decreasing until t_m, the r.h.s. of the last inequality is strictly larger than $g(\tilde{M} - 1) - g(\tilde{M}) - 2\epsilon$. By taking ε sufficiently small, we get $g_k(t) - g_k(\tilde{M}) > 0$ for any $t < \tilde{M}$. Hence, with a positive probability, we get eventually $\hat{\sigma}_k \geq \tilde{M}$.

If $M \leq t_m$, then $\tilde{M} = M$ and with positive probability eventually we have $\hat{\sigma}_k \geq M$. Since $\hat{\sigma}_k \leq T_k \leq \sup_k T_k = M$, we have eventually $\hat{\sigma}_k = T_k = M$. For one of such k, it holds $\frac{\hat{\sigma}_k}{T_k} = 1 > \lambda$, so that, by the definition of the RP, at the following iteration with positive probability we have $T_{k+1} > T_k = M = \sup_k T_k$, which is impossible.

In the other case, $t_m < M \leq \frac{t_m}{\lambda}$, we have $\tilde{M} = t_m$, and there is a positive probability that $t_m = \tilde{M} \leq \hat{\sigma}_k \leq T_k \leq \sup_k T_k = M$ for k large enough; for any of these values of k, we get $\frac{\hat{\sigma}_k}{T_k} \geq \frac{t_m}{T_k} \geq \frac{t_m}{M} \geq \lambda$. As a consequence, with positive probability eventually we have $T_{k+1} = f_T(T_k) > T_k$, which is a contradiction with $\sup_k T_k = M$. ■

Theorem 3.2. *If we define* $T := \left\lceil \frac{t_m}{\lambda} \right\rceil$, *then*

1. $\mathrm{P}\left(\bigcap_{t=1}^{T} \lim_{k \to \infty} \hat{p}_k(t) = p(t) \right) = 1,$
2. $\mathrm{P}\left(\lim_{k \to \infty} \hat{\sigma}_k = t_m \right) = 1.$

Proof. **1**. By theorem 3.1, for any $t = 1, 2, \ldots, T$, with probability one we can eventually compute $\hat{p}_k(t)$. Hence, by the two statements of lemma 3.1 and the strong law of large numbers, we get

$$\mathrm{P}\left(\bigcap_{t=1}^{T} \lim_{k \to \infty} \hat{p}_k(t) = p(t) \right) = 1,$$

that completes the proof of this point.
2. From 1 and the continuous mapping, with probability one, we have

$$\lim_{k \to \infty} g_k(t) = g(t),$$

for $t = 1, \ldots, T$, with $T > t_m$. Therefore, the sequence $\hat{\sigma}_k$ converges to t_m. ■

Remark 3.2. The efficiency of the RP depends on the expected value of the ratio $\sup_k T_k/t_m$. Although we do not have derived upper-bounds for this ratio, in all applications we performed, it remains sufficiently close to one.

4. Numerical Results

Below, we describe some results of the application of the RP to solve different instances of the TSP studied in [18] and two instances of a pseudo-Boolean problem. The underlying algorithm used here in the RP is mainly the ACO proposed in [18], known as MMAS; for the TSP instances, it is combined with different local search procedures [18]. In addition, for the pseudo-Boolean problem, we also use a GA with a population of 20 individuals. At each GA iteration, first the individuals are ordered accordingly to the value of the fitness function. Then, the first half individuals of the list are kept. The remaining individuals are replaced by new ones, obtained from the first group in the following way: we first draw without replacement

random pairs of individuals; for each pair, two new individuals are produced by using single-point crossover. The new population is finally obtained by flipping a component of the binary string chosen uniformly and independently for each individual, in the case that this decreases the value of the fitness function.

The RP setting is as follows: $r_{k+1} = f_r(r_k) := c_1 \cdot r_k$ and $T_{k+1} = f_T(T_k) := c_2 \cdot T_k$ where $c_1 = 1.2, c_2 = 1.1$. The initial values for r_0 and T_0 are 20 and 100, respectively. Finally, we set $\lambda = \frac{4}{5}$.

For both the TSP instances and the pseudo-Boolean problem considered here, the optimal solution is known. This information can be used to estimate the failure probability of the RP and of the underlying algorithm. However, obviously this information cannot be used when applying the RP.

In order to compare the results from the two algorithms with the same computational effort, we consider for the RP a pseudo-time *t*, defined as follows: for the initial RP iteration, the first T_0 instants of the pseudo-time correspond to the first T_0 iterations of the first replication of the underlying algorithm; the following T_0 pseudo-time instants correspond to the first T_0 iterations of the second replication and so on until replication r_0. At the end of the *k*-th RP iteration, we have produced r_k executions (replications) for T_k times and the final pseudo-time instant is $t = r_k \cdot T_k$. At the $(k+1)$-th iteration, we have a certain (r_{k+1}, T_{k+1}), with either $r_{k+1} > r_k$ and $T_{k+1} = T_k$ or $r_{k+1} = r_k$ and $T_{k+1} > T_k$. In the first case, the pseudo-time instant $t = T_k \cdot r_k + 1$ corresponds to the first iteration time of the $r_k + 1$ replication and it is increased until the end of that replication. We proceed in the same way until the end of r_{k+1} replication. In the second case, the pseudo-time instant $t = T_k \cdot r_k + 1$ corresponds to the iteration time $T_k + 1$ of the first replication and is then increased until the iteration time T_{k+1} of that replication. Then, the same procedure is applied for the remaining replications based on their number.

We denote by $\bar{Y}(t)$ $(t = 1, 2, \ldots)$ the process describing the best so far solution of the RP (MMAS or GA) corresponding to the pseudo-time (time) instant *t*. Hence, based on a set of *m* independent replications of the RP, we can estimate the failure probability $p_{\mathrm{RP}}(t)$ by using the classical estimator

$$\hat{p}_{\mathrm{RP}}(t) = \frac{1}{m} \sum_{i=1}^{m} 1_{\{\bar{Y}_i(t) > f_m\}}, \tag{5}$$

and analogously with $\hat{p}(t)$ for the MMAS or GA. By the law of large numbers this estimator converges to the failure probability $p_{\mathrm{RP}}(t)$ (to $p(t)$ for the MMAS or GA).

We start with the example where we want to minimize the following pseudo-Boolean function

$$f(x) = -\left| \sum_{i=1}^{N} x_i - \frac{N-1}{2} \right|, \tag{6}$$

with respect to all binary strings of length *N*. In Figure 1, this function is plotted versus the number of 1s in the case of $N = 50$ considered now.

This function has two local minima but only one of them is global. The first base algorithm considered is the MMAS, for which the pheromone bounds τ_{min} and $\tau_{\max}$ ensure that at any time, there is a positive probability to visit each configuration, e.g. the global minimum.Therefore, with probability one this algorithm will find the solution. However, if it

reaches a configuration with few 1s, it takes in average an enormous amount of time, not available in practice, to move towards the global minimum.

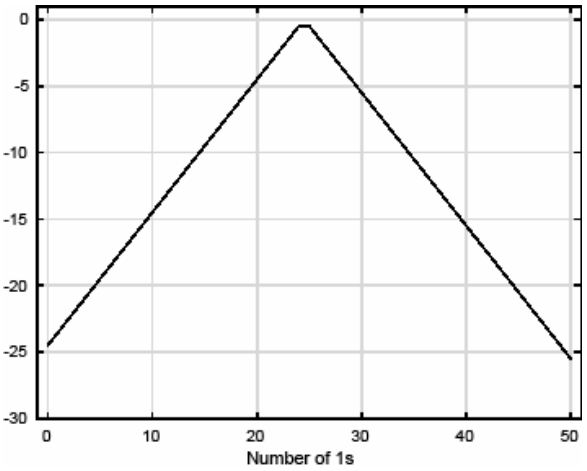

Figure 1: Plot of the considered pseudo-Boolean function value versus the number of 1s of the binary string of length $N = 50$.

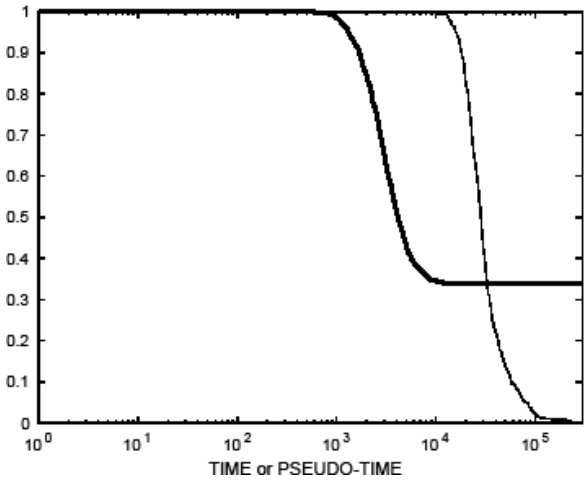

Figure 2: Pseudo-Boolean problem instance with $N = 50$ (Boolean 50) solved by the RP applied to MMAS (a single ant is used and we set $\tau_{\min} = 0.4$, $\tau_{\max} = 1 - \tau_{\min}$, and $\rho = .01$).The estimated f.p. as function of pseudo -time or time for the RP (thin line) and the MMAS (thick line) are computed by the estimator in (5) based on 500 and 1000 replications, respectively.

Therefore, we expect that in this case the restart will be successful.

In Figure 2, we show the estimated failure probability (f.p.) $\hat{p}(t)$ for the MMAS algorithm to minimize the pseudo-boolean function of Figure 1 (thick line). In the same figure, the estimated f.p. $\hat{p}_{\mathrm{RP}}(t)$ of the RP is plotted versus the pseudo-time (thin line). We notice that there is a clear advantage to use the RP when compared to the standard MMAS.

For this problem, we also use the GA described before. In this case we have $N = 300$ (boolean300). The curves obtained are very similar to those in Figure 2. The f.p. estimated values are shown in Table 1.

We consider now an instance of the TSP with 532 cities (att532) solved by MMAS with the same settings as in [18]. After five hundreds of thousands of iterations, the underlying algorithm has an estimated f.p. of 0.38 ca. Instead, at the same value of the pseudo-time, the RP has a significantly lower f.p. (0.004 ca), as clearly shown in Fig. 3. We remark that, until the value 3900 ca for the time or pseudo-time, the f.p. of the underlying algorithm is lower than the one of the RP. This is due to the fact that the RP is still learning the optimal value of the restart time. After that, the trend is inverted: the RP overcomes the MMAS and gains two orders of magnitude for very large values of the pseudo-time.

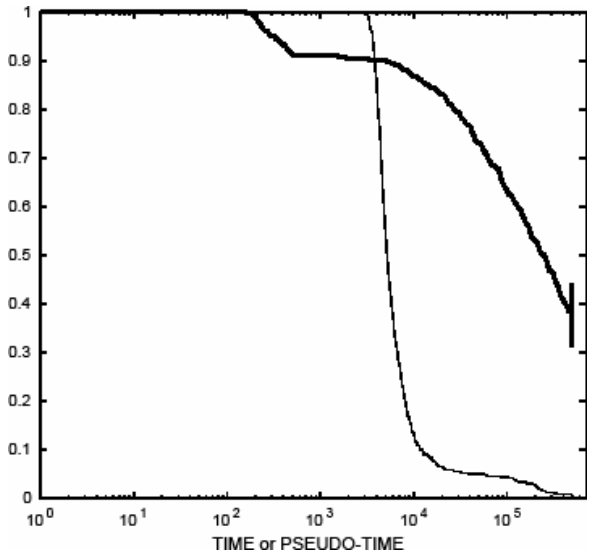

Figure 3: The TSP instance with 532 cities (att 532) solved by the RP applied to MMAS (settings as in [18]). The estimated f.p. as function of pseudo-time or time for the RP (thin line) and the MMAS (thick line) are computed by the estimator in (5) based both on 500 replications. The vertical segment shows the interval for 99% confidence level.

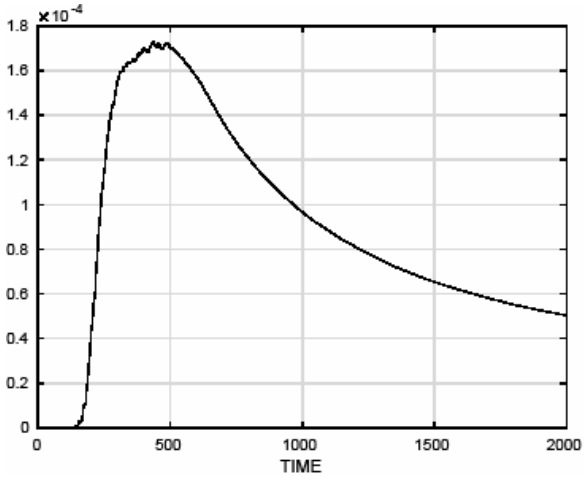

Figure 4:The TSP instance with 532 cities (att 532) solved by the RP applied to MM AS.The denominator of the function $g_k(t)$ at the end of a single RP replication.

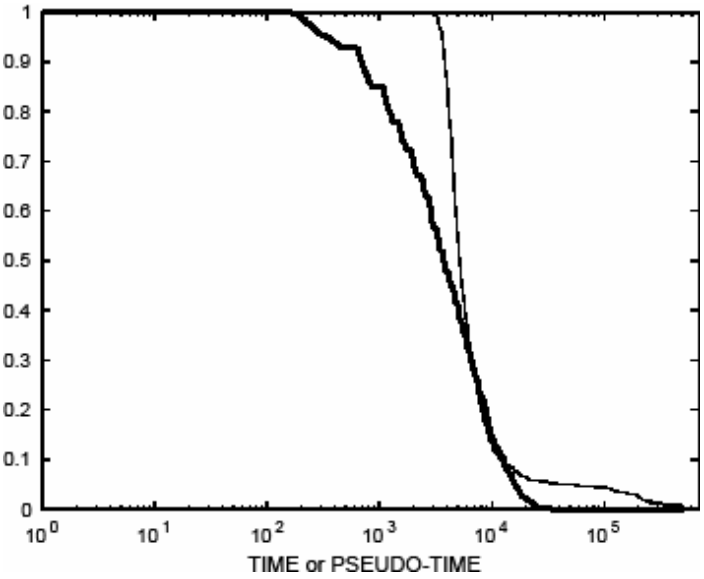

Figure 5: The TSP instance with 532 cities (att 532) solved by the RP applied to MMAS. Comparison between the failure probability curve of the RP that appears in Fig. 3 (thin line) and the one obtained by applying the standard restart with the optimal restart time (thick line). The f.p. curves of both the RP and the standard restart are computed by the estimator in (5) based on 500 replications.

We notice that the value $\hat{\sigma}_k$ approaches the optimal restart time t_m. In fact, as an example, in Figure 4, we show the denominator of the function $g_k(t)$ at the end of a single RP execution. A global maximum appears at approximately the value of 430, the difference with the value of t_m, computed from the estimate $\hat{p}(t)$, being less than 1%.

Finally in Figure 5, we compare the f.p. curve for the RP with the one obtained applying the standard restart with the estimated optimal restart time. We remark that to use this approach in practice would require much longer computation than to execute the RP. In fact, we need to provide first an estimate of t_m by means of a sufficiently large sample of independent runs of the base algorithm.

We notice that the RP curve starts to decrease significantly after the other one. This is due to the fact that the RP is still searching for the optimal value of the restart, whereas it is set from the beginning in the other (ideal) case. At about pseudo-time 7000, the two f.p. s become almost equal. After that, the f.p. of the standard restart goes to zero faster, even if the difference between the two f.p. s remains less than 0.05 ca. Finally, at pseudo-time $5 \cdot 10^5$, the estimated f.p. value of the RP is $4 \cdot 10^{-3}$, comparable to the estimated f.p. value of the standard restart at same time.

We notice that curves similar to those as in Figures 3, 4 and 5 were obtained for all the other TSP instances considered. The corresponding results are shown in Table 1.

By looking at the results in Table 1, it is evident the advantage of using the RP instead of the underlying algorithm. In fact, for each case, the estimated f.p. value of the RP is several orders of magnitude lower than the one of the base MHA.

Table 1: Results for the application of the RP and the base MHA to TSP Instances with known optimal solution and the instances of a pseudo-Boolean problem. The estimated f.p. values are computed at the time Tc reported in the third column (pseudo-time for the RP) based on at least 500 elements.

Instance	**Base MHA**	**Tc**	**Base MHA f.p.**	**RP f.p.**
Boolean50	MMAS	300000	0.34	$2.1 \cdot 10^{-3}$
Boolean300	GA	10000	0.43	0
pcb442	MMAS·3opt	100000	0.22	$4.0 \cdot 10^{-3}$
att532	MMAS·3opt	500000	0.38	$4.0 \cdot 10^{-3}$
lin318	MMAS·2.5opt	30000	0.44	0
d1291	MMAS·3opt	700000	0.57	$2.0 \cdot 10^{-3}$
d198	MMAS·2.5opt	100000	0.67	0

5. Conclusions

Given a combinatorial optimization problem, it is often needed to apply stochastic

algorithms exploring the space using a general criterion independent of the problem. Unfortunately, usually there is a positive probability that the algorithm remains in a sub-optimal solution. This drawback can be coped by applying periodic algorithm re-initializations. This strategy is called restart. Although it is often applied in practice, there are few works studying it theoretically. In particular, there are no theoretical information to be used in practice to choose a convenient value for the restart time.

In this paper, we propose a new procedure to optimize the restart (RP) and we study it theoretically. The iterative procedure starts by executing a certain number r_0 of independent replications of the underlying algorithm for a predefined time T_0. At the end of any iteration k of the RP, we have r_k independent replications each composed by T_k iterations of the underlying algorithm. We then compute the minimum value $\tilde{Y}_k$ of the objective function $f(\cdot)$ on these $r_k T_k$ points. Hence, for each time $t = 1, \ldots, T_k$, we compute an estimate $\hat{p}_k(t)$ of the failure probability $p(t)$, i.e. the probability that we have not yet reached the value $\tilde{Y}_k$. We now consider the function $g_k(t) = [(1 - \hat{p}_k(t)^{\frac{1}{t}})\hat{p}_k(t)]^{-1}$, that is the analogous of the function $g(t) = [(1 - p(t)^{\frac{1}{t}})p(t)]^{-1}$. We recall that the first time t_m at which the absolute minimum of $g(t)$ is reached corresponds to an "optimal value" of the restart time, that minimizes the expected time to find a solution. We then compute the position $\hat{\sigma}_k$ of the first minimum of $g_k(t)$. If $\hat{\sigma}_k$ is close to the end of the current execution time frame T_k of the underlying algorithm, T_k is increased; otherwise this is done for the number of replications r_k. This is controlled by the parameter $\lambda \in (0,1)$.

The theory predicts that the RP eventually will find the optimal value of the restart time. In fact, the theorems prove that, with probability one, $\hat{p}_k(t)$, $g_k(t)$ and $\hat{\sigma}_k$ converge to $p(t)$, $g(t)$ and t_m, respectively.

In this paper, we illustrate some results obtained by applying the RP to two versions of the MMAS ACO algorithm [18] for solving several TSP instances, whose solution is known, with hundreds or thousands of cities. We also present some results using the RP with MMAS and a GA for solving two instances of a pseudo-Boolean problem. The results obtained show that the estimated values of the failure probability of the RP are several orders of magnitude lower than those of the underlying algorithms, for equal computational cost. Therefore, given a certain computation resource, by applying the RP, we are far more confident that the result obtained is a solution of the COP instance analyzed. The procedure proposed could be improved preserving its performance and decreasing the computational cost. A possible way to do it is to increase the parameter λ along iterations. In fact, once we have a reasonably good estimate of $g(t)$, we would like to reduce the possibility that, by chance, we increase too much the time interval length. This can be done by increasing the value of λ.

Acknowledgments
The authors are very thankful to Professor Mauro Piccioni for his very useful comments and suggestions and to Professor Thomas Stützle for the ACOTSP code.

References

[1] D. L. Applegate, R. M. Bixby, V. Chvátal, and W. J. Cook, *The Traveling Salesman Problem*, Princeton University Press, Princeton, 2006.

[2] C. Blum, and A. Roli, Metaheuristics in combinatorial optimization: overview and

conceptual comparison, *ACM Computing Surveys* **35**(3), (2003), 268–308.

[3] L. Carvelli, and G. Sebastiani, Some issues of aco algorithm convergence, In: *Ant Colony Optimization - Methods and Applications*, A. Ostfeld, ed. IntechOpen, (2011), 39–52.

[4] M. Dorigo, and T. Stützle, *Ant Colony Optimization*, MIT Press,Massachucetts, 2004.

[5] M. Garvels, and D. Kroese, A comparison of restart implementations, In: *WSC'98 Proceedings of the 30th Conference on Winter Simulation*, IEEE Computer Society Press, CA, (1998), 601–608.

[6] ——, On the entrance distribution in restart simulation, In: *RESIM '99 Proceedings of the Rare Event Simulation Workshop*, University of Twente, The Netherlands, (1999), 65–88.

[7] S. Geman, and D. Geman, Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **6**(6), (1984), 721–741.

[8] F. Glover, Tabu search-part 1, *ORSA Journal of Computing* **1**(3), (1989), 190–206.

[9] D. Goldberg, B. Korb, and K. Deb, Messy genetic algorithms: motivation, analysis, and first results, *Complex Systems* **3**, (1989), 493–530.

[10] W. Gutjhar, A generalized convergence result for the graph-based ant system, *Probability in the Engineering and Informational Sciences* **17**, (2003), 545–569.

[11] W. Gutjhar, and G. Sebastiani, Runtime analysis of ant colony optimization with best-so-far reinforcement, *Methodology and Computing in Applied Probability* **10**, (2008), 409–433.

[12] T. Hansen, On the analysis of dynamic restart strategies for evolutionary algorithms, In: *Proceedings of the 7th International Conference on Parallel Problem Solving from Nature*, London, Springer-Verlag, (2002), 33–43.

[13] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, Optimization by simulated annealing, *Science* **220**(4598), (1983), 671–680.

[14] A. Misevicius, Restart-based genetic algorithm for the quadratic assignment problem, In: *Research and Development in Intelligent Systems XXV*, M. Bramer, M. Petridis, and F. Coenen, eds. Springer-Verlag, Berlin-Hidelberg, (2009), 91–104.

[15] F. Neumann, and C. Witt, Runtime analysis of a simple ant colony optimization algorithm, *Algorithmica* **54**, (2007), 243–255.

[16] A. Van Moorsel, and K. Wolter, Analysis and algorithms for restart, In: *QEST '04*

Proceedings of the 1st International Conference on the Quantitative Evaluation of Systems, Munich, Germany, (2004), 195–204.

[17] L. T. Schmitt, Theory of genetic algorithms, *Theoretical Computer Science* **259**, (2001), 1–61.

[18] T. Stützle, and H. Hoos, Max-min ant system, *Future Generation Computer Systems* **16**, (2000), 889–914.

_____

Article history: Submitted November, 23, 2018; Accepted January, 04, 2019.