Acceptance Driven Local Search and Evolutionary Algorithms

Eric Poupaert and Yves Deville*
Department of Computing Science and Engineering
Université catholique de Louvain
Place Sainte Barbe 2, B-1348 Louvain-la-Neuve, Belgium
Email: {ep.yde}@info.ucl.ac.be
Tel: +32 10 473150

Abstract

Local Search (LS) and Evolutionary Algorithms (EA) are probabilistic search algorithms, widely used in global optimization, where selection is important as it drives the search. In this paper, we introduce *acceptance*, a metric measuring the selective pressure in LS and EA, that is the trade-off between exploration and exploitation. Informally, acceptance is the proportion of accepted non-improving transitions in a selection.

We propose a new LS algorithm, SAad, based on acceptance schedule (a schedule for the selective pressure). In EA, two new selection rules based on the Metropolis criterion are introduced. They allow two new EA (2MT and RT) based on acceptance schedule. They demonstrate a possible way of merging LS and EA technologies. Benchmarks show that the developed algorithms are more performant than standard SA and EA algorithms, and that SAad is as efficient as the best SA algorithms while 2MT and RT are complementary to Evolution Strategies.

1 Introduction

Local Search (LS) and Evolutionary Algorithms (EA) are probabilistic search algorithms widely used in global optimization. Such problems can be formalized as a set of solutions (called search space) and a function evaluating the solutions (called score, energy or fitness). The aim of global optimization is to find a solution such that no other solution is better. LS is based on the concept of neighborhood; its principle is to improve iteratively a current solution by generating and selecting neighbor solutions. The principle of EA is to model the evolution of a population of individuals through recombination, mutation and selection.

Selection is an important part of both LS and EA: it drives the search toward promising zones of the search space. Selection is subject to an important trade-off: it either favors the *exploration* of the search space or the *exploitation* of the neighborhood (or population). This trade-off is usually expressed using the informal term of *selective pressure*: high pressure implying exploitation and low pressure exploration. Measuring selective pressure is an important trend in EA. Takeover time, for example, is the metric used in (Bäck, 1994).

A contribution of this paper is to provide a metric measuring the selective pressure appropriate for both LS and EA. We will therefore introduce the notion of *acceptance*, the proportion of accepted non-improving transitions in a selection.

The best-known EA are: Genetic Algorithms (GA, Holland, 1975), Evolution Strategies (ES, Bäck, 1996) and Evolutionary Programming (EP, Fogel, 1992).

In the vast majority of EA, selection does not vary during the search. However, a varying selection parameter is used in an example in (Davis, 1991), and a Boltzmann Tournament Selection aiming at a niching mechanism is described in (Goldberg, 1990).

Simulated Annealing (SA, Kirkpatrick et al., 1983) is a LS algorithm originating from a simulation of the thermodynamics of gas. In SA, selection is performed using the Metropolis criterion (Metropolis et al., 1955) which has a parameter called temperature. The principle of SA is to enforce a temperature schedule reducing temperature progressively during optimization. This reduction of temperature permits to achieve a convergence to the global optimum. The temperature schedule is critical to the success of SA.

The best-known temperature schedule SAgeo is the geometric one (Kirkpatrick et al., 1983). Another important one is SApoly (van Laarhoven, 1988) where temperature reduction is performed using a feedback mechanism based on the concept of quasi-equilibrium.

*This research is partially supported by the *actions de recherche concertée* ARC/95/00-187.

It has a polynomial time complexity. In addition to an analog feedback, SAEF innovates by introducing a variable chain length based on an approximate measure of equilibrium. It is considered as one of the best SA known of to date (Aarts and Lenstra, 1997).

In SA, temperature has a direct impact on the selective pressure which initially is low and increases with time. The progressive increase of selective pressure is the core of of SA. In LS and in EA, although selection may vary during the search, the resulting selective pressures can only de deduced during the execution. In existing algorithms, selection is not adapted according to a given schedule for the selective pressure.

A contribution of this paper is the definition of *acceptance schedule* (a schedule for the selective pressure) and an associated algorithm (ESTIMATE_PARAMETER) computing the successive values of a selection parameter (temperature) in order to achieve an acceptance schedule. Since acceptance is appropriate for both LS and EA, it *makes possible the merging of SA and EA technologies.*

The other contributions of this paper are now described. *We designed and implemented a new local search algorithm*, SAAD, based on acceptance schedule. Its temporal complexity can be *a priori* computed and is $O(v \log v)$ (where v is the average size of a neighborhood). Benchmarks have shown that SAAD is more performant than standard SA techniques, and is as efficient as the best SA algorithms. *We defined two selection rules*, a relaxed 2-Tournament and a relaxed truncation, applicable in EA. These rules introduce the Metropolis criterion on populations and allow for adaptable acceptance. They respect the design guidelines expressed in (Bäck, 1994). *We designed and implemented new evolutionary algorithms*, 2MT and RT, based on acceptance schedule and implementing the two selection rules. Benchmarks have shown they are more performant than standard EA, and are complementary to ES.

The paper is structured as follows. In Section 2, acceptance driven SA is presented; acceptance and acceptance schedule are defined and algorithm SAAD is described. Section 3 presents acceptance driven EA; two selection rules are proposed and algorithms EAAD is described. Experimental results are analyzed in Section 4.

2 Acceptance Driven SA

2.1 Definition of Acceptance

A transition occur when the current solution is replaced by one of its neighbors. Transitions that improve the current solutions are natural since they contribute to both the exploration of the search space and the exploitation of the neighborhood. Non improving transitions go in the direction of exploration but to the detriment of exploitation. Selective pressure is related to the probability that these transitions occur.

We propose a new metric of selection pressure, called *acceptance*. Intuitively, it is the proportion of non-improving transitions that are accepted. Moreover, it is a global measure of the solution space and is not relative to a specific current solution.

Definition: Given a local search algorithm LS using a selection rule SELECT, a neighbor function NEIGHBOR and an energy function ENERGY,

$$\textit{acceptance} = \mathcal{P}(\text{SELECT}(S, S', t) = S' \mid \text{ENERGY}(S') > \text{ENERGY}(S) \;\&\; S' = \text{NEIGHBOR}(S))$$

where S and S' are solutions and t a parameter of the selection. The upper bound 1 of acceptance implies that non improving transitions are always accepted; and the lower bound 0 that they are never accepted. Acceptance is relative to a selection rule SELECT and its parameter t. As t may vary during the search, such as in SA, acceptance may also vary.

2.2 Local Search Driven by Acceptance

```
S := INITIAL_SOLUTION
s := 0
while CONTINUE do
    χ := TARGET_ACCEPTANCE(s)
    t := ESTIMATE_PARAMETER(χ)
    repeat L times
        S' := NEIGHBOR (S)
        S := SELECT (S,S',t)
    end
    s := s + 1
end
return S
```

Algorithm 1: Acceptance driven EA : SAAD

Temperature reduction is the core of SA. In Algorithm 1, we propose a new LS algorithm, called Simulated Annealing Driven by Acceptance (SAAD), based on SA with an acceptance schedule. To this end, a parameter χ and an index s are used.

Selection is performed in SA using the Metropolis criterion (Algorithm 2). Its parameter t controls the selective pressure. Hence

$$\text{Select}(S, S', t) \Leftrightarrow \text{Metropolis}(S, S', t)$$

The relationship between acceptance and the selection parameter (temperature in SA) is performed by the ESTIMATE_PARAMETER function specified hereafter. The successive iterations where the parameter is kept constant is called a *chain*.

function $S'' = $ METROPOLIS(S,S',t)
begin
  $\Delta :=$ ENERGY(S') $-$ ENERGY(S)
  $p := \min(1, \exp(-\Delta/t))$
  if RANDOM$(0,1) < p$ **then** $S'' := S'$ **else** $S'' := S$
end

Algorithm 2: Metropolis criterion

function $t = $ ESTIMATE_PARAMETER (χ)
Pre: $\chi \in [0,1]$
Post: $t \geq 0$ such that the expected acceptance is equal to χ for a chain using t as the value of the parameter of SELECT .

An *acceptance schedule* is a method to determine, for each moment of the search, an acceptance we would like to enforce (called *target acceptance*). This is the role of TARGET_ACCEPTANCE.

function $\chi = $ TARGET_ACCEPTANCE(s)
Pre: $s \geq 0$ is the index of a chain
Post: $\chi \in [0,1]$ is the target acceptance for the chain of index s

2.3 Acceptance Schedule

The initial value χ_0 of acceptance, $\chi_0 = 1$, is its upper bound. It leads to a complete coverage of the search space.

The next step is to determine the decrease of acceptance. A well-known heuristic used in simulated annealing states that "the number of (accepted) transitions must be constant for each chain". This implies that chains of lower temperature (and thus lower acceptance) must be longer. We have modified this heuristic to fit our framework: "the number of (accepted) transitions per unit of acceptance is constant", that is $\chi(s).L/(\chi(s+1)-\chi(s))$ (where s is the chain index) is constant. This implies that more time must be spent for lower acceptance. This leads to a differential equation whose solutions are: $\chi(s) = \alpha \cdot \exp(-\beta \cdot s)$. We have also $\alpha = \chi(0) = 1$. As $\chi(s)$ forms a geometric sequence, it can be expressed in term of half-life (denoted $s_{1/2}$): the number of chains such that the acceptance is divided by two (i.e. $\chi(s + s_{1/2}) = \chi(s)/2$, where $s_{1/2}$ is an input parameter of the algorithm). We obtain finally:

$$\text{TARGET\_ACCEPTANCE}(s) = \chi := (0.5)^{s/s_{1/2}} \qquad (1)$$

Since we want acceptance to decrease with time, we have $s_{1/2} > 0$. A higher value of $s_{1/2}$ corresponds to a slower decrease of χ.

The stopping condition is traditionally seen as being part of a schedule. The criterion usually used in local search is to stop when no further improvement of the current solution is to be expected. In our implementation, we have chosen to stop when the expected number of non improving transitions during *stop* (an input parameter of the algorithm) consecutive chains is below 1/2:

$$\text{CONTINUE} \Leftrightarrow stop \cdot L \cdot \chi \geq 1/2 \qquad (2)$$

In practice, L is set to 3 times the size of the neighborhood, *stop* between 5 and 10 and $s_{1/2}$ is set according to the time available for optimization.

2.4 Estimation of the Selection Parameter

At the beginning of each chain, the parameter t has to be estimated such that the expected acceptance over this chain is equal to the target acceptance χ. This estimation uses a feedback mechanism. Given a transition $S \rightarrow S'$ and the value of t, the probability that this transition is accepted can be computed from the SELECT function. Likewise, the acceptance over a chain can be computed *a posteriori* knowing the transitions proposed by NEIGHBOR.

Let $acc(H,t)$ be a measure of the acceptance, called *acceptance function*, over a chain where H is the set of the (proposed) transitions. We have:

$$acc(H,t) = \mathcal{P}(\text{SELECT}(S,S',t) = S' \mid \text{ENERGY}(S') > \text{ENERGY}(S) \;\&\; (S \rightarrow S') \in H)$$

Estimating t_i for the chain of index i is therefore equivalent to find t_i such that $acc(H_i,t_i) = \chi_i$.

Unfortunately, H_i is not known *a priori*; especially since H_i results from a stochastic process involving the value of t_i. A way to solve this problem is to suppose that $acc(H_i,t_i) \simeq acc(H_{i-1},t_i)$ because H_{i-1} will be known when t_i will have to be computed. This hypothesis is realistic when t_i and t_{i-1} are not too distant and when H_{i-1} is large enough to be a good representative of the transitions that *could* have been proposed during this chain. From a statistical point of view, it is the case when L is of the same order as the size of the neighborhood.

Estimating t_i is thus equivalent to solving $acc(H_{i-1},t_i) = \chi_i$, where H_{i-1} and χ_i are known.

Given that selection is based on the Metropolis criterion, the acceptance function can easily be computed from a set H of non improving transitions.

$$acc(H,t) = (1/n) \cdot \sum_{j=0}^{n-1} \exp(-\Delta_j/t) \qquad (3)$$

where $\Delta_j > 0$ is the energy difference of the jth transition of H and n the size of H.

In order to implement the ESTIMATE_PARAMETER function, the acceptance function must be inverted; this is possible as $acc(H,t)$ is monotonous relatively to t. To this end, a Newton-Raphson (N-R) method can be used. For numerical stability reasons, we preferred to use a variation of N-R, working on the logarithm of the parameter. The principle is to find a root of a function by forming a sequence of better and better estimates of the root. Suppose that we want to find the root of the equation $f(x) = 0$ and that we have an initial estimate x_0 of a solution, the (variant of) N-R sequence is:

$$x_{k+1} = x_k \cdot \exp\left(\frac{-f(x_k)}{f'(x_k)/x_k}\right) \tag{4}$$

where f' is the derivative of f. If the sequence converges, its limit is a solution of the equation.

In our case, the equation $acc(H,t) - \chi = 0$ must be solved. We therefore have:

$$t_{k+1} = t_k \cdot \exp\left(\frac{n \cdot \chi - \sum_{j=0}^{n-1} \exp(-\Delta_j/t_k)}{\sum_{j=0}^{n-1} (\Delta_j/t_k) \cdot \exp(-\Delta_j/t_k)}\right) \tag{5}$$

It is evidenced by experiments that if the temperature of the previous chain is used as initial estimate of the one of the next chain, a single step of N-R leads to an appropriate precision.

Without initial estimate, t can be approximated as

$$t \approx -\overline{\Delta}/\ln(\chi) \tag{6}$$

where $\overline{\Delta}$ is the average of the various Δ_j. This estimation is only accurate when $\chi > 0.9$.

The proposed algorithm must now be accommodated to maintain a set of transitions:

```
1.1   H := ∅
5.1   t := ESTIMATE_PARAMETER(H,χ)
5.2   H := ∅
8.1   if ENERGY(S') > ENERGY(S)
8.2       then H := H ∪ {S → S'}
8.3   S := SELECT (S,S',t)
```

In practice, each time a non improving transition is proposed, the contribution of the corresponding Δ to the sums of Eqs. 5 and 6 are accumulated. Therefore, H is not stored as an explicit set of transitions but as quadruplet (H_0, H_1, H_2, H_3) where:

H_0 = n (number of transitions, see Eqs. 5 and 6)
H_1 = sum of $\exp(-\Delta_j/t)$ (see Eq. 5)
H_2 = sum of $(\Delta_j/t) * \exp(-\Delta_j/t)$ (see Eq. 5)
H_3 = sum of Δ_j/t (see Eq. 6)

These variables can be updated by simple instructions replacing Instruction 8.2. as the transitions themselves are now useless. An implementation of the ESTIMATE_PARAMETER function is given in Algorithm 3.

2.5 Complexity

Let *ep* be the complexity of ESTIMATE_PARAMETER, *ng* of NEIGHBOR, *sel* of SELECT, *en* of ENERGY and s_{max} the total number of chains. The complexity of the algorithm SAAD is:

$$O(s_{max} \cdot ep + s_{max} \cdot L \cdot (ng + sel))$$

```
function t = ESTIMATE_PARAMETER(H,χ,t_old)
   begin
      if H_0 = 0 then t := ∞
      else if χ > 0.9 then t := −H_3/(H_0 · ln(χ))
      else t := t_old · exp((H_0 · χ − H_1)/H_2)
   end
```

Algorithm 3: ESTIMATE_PARAMETER

The value of s_{max} can be derived from the acceptance schedule (Eq. 1) and the stopping condition (Eq. 2):

$$s_{max} = s_{1/2} \cdot (1 + \log_2(L \cdot stop))$$

Moreover $ep = O(1)$. The *stop* parameter is problem independent and can thus be seen as a constant (fixed between 5 and 10 in our implementation). Using Metropolis, we have $sel = O(en)$. For most problems $ng = O(1)$ (generation and choice of a neighbor), and $en = O(1)$ as the energy can be computed incrementally within the NEIGHBOR function. As already justified, we fixed L to $O(v)$, where v is the average size of a neighborhood. The complexity of Algorithm SAAD becomes finally

$$O(s_{1/2} \cdot v \cdot \log(v)) \tag{7}$$

The space complexity of SAAD is $O(L \cdot \text{size}(S))$ but is reduced to $O(1)$ using the proposed implementation of the acceptance function.

It is noteworthy that the temporal complexity of SAAD, which also is the total number of generated neighbors $(L \cdot s_{max})$, can be computed *a priori*. This is usually not the case for classical SA algorithms, where a temporal complexity in often difficult to derive. In (van Laarhoven and Aarts, 1987), it is shown that for specific decrement rules and stop criteria, the total number of generated neighbors is $O(v \log(q))$, where q is the size of the set of configurations (usually exponential).

3 Acceptance Driven EA

In LS, the current state is the so-called current solution; in EA, it is a population. The evolution of this current population is performed by generating a second population (called offspring of the first) through the MUTATE and RECOMBINE functions, and, selecting individuals within these populations.

Our acceptance driven EA, called EAAD, is given in Algorithm 4. It is obtained by adapting SAAD (Alg. 1) to fit in a standard EA scheme, where P denotes the current population and contains n individuals, and P' a population offspring from P and contains m individuals. The role of SELECT is reflected in the following specification:

```
P := array: [1..n] → Individual
for each i ∈ [1..n]: P[i] := RANDOM_INDIVIDUAL
s := 0
while CONTINUE do
    χ := TARGET_ACCEPTANCE(s)
    t := ESTIMATE_PARAMETER(χ)
    repeat L/m times
        P' := MUTATE(RECOMBINE(P))
        P := SELECT (P, P', t)
    s := s + 1
return BEST_OF (P)
```

Algorithm 4: Acceptance driven EA (EAAD)

function $P'' = \text{SELECT}\ (P,\ P',\ t)$
Post: $P'' \subseteq P \cup P'$ and $\#P'' = \#P$
Note: t has an impact on the selective pressure

In EAAD, the length of the chains is L/m, where m is the size of population P'. Hence, in terms of number of individuals to be evaluated, the length of the chains is L, as in SAAD.

The CONTINUE and TARGET_ACCEPTANCE functions can be implemented as in SAAD. To complete the EAAD algorithm, the ESTIMATE_PARAMETER and the SELECT functions have to be implemented.

3.1 Acceptance within Populations

As selections are performed on populations, the definition of acceptance given in Section 2 must be generalized. It is convenient to introduce a reference selection rule (SELECT_REF) stating which transitions should be accepted and which ones should be rejected using an exploitation oriented view. It thus has no third parameter.

Let $P'' = \text{SELECT}(P, P', t)$ (or $P'' = \text{SELECT\_REF}(P, P')$), and let $S \to S'$ be a transition with $(S, S') \in P \times P'$. This transition is *accepted* if $S \notin P''$ (S was in P but no longer in P''), and $S' \in P''$ (S' is selected in P'' from P'). This transition is *rejected* if $S \in P''$ (S was in P and is kept in P'') and $S' \notin P''$ (S' was a potentially new candidate from P', but is not selected in P''). The other possible transitions are meaningless and are thus neither accepted nor rejected.

If we take a 2-Tournament as SELECT_REF, a transition $S \to S'$ is rejected if it is non improving. In this case, each transition is evaluated separately.

On the other hand, if Truncation is used as SELECT_REF, the transitions are evaluated globally. A transition $S \to S'$ is rejected if S is in the set of the n best individuals of $P \cup P'$ and S' is not in this set (n is the size of P).

Definition: Given an evolutionary algorithm EAAD and a function SELECT_REF, the *acceptance* of EAAD is the expected proportion of transitions generated using RECOMBINE and MUTATE and rejected by SELECT_REF that are accepted by SELECT. Formally:

$$
\begin{aligned}
&\mathit{acceptance} = \mathcal{P}(S \notin P''\ \&\ S' \in P''\ |\\
&\quad P' = \text{MUTATE}(\text{RECOMBINE}(P))\\
&\quad \&\ P'' = \text{SELECT}(P, P', t)\ \&\ P''' = \text{SELECT\_REF}(P, P')\\
&\quad \&\ (S, S') \in P \times P'\ \&\ S \in P'''\ \&\ S' \notin P'''\ )
\end{aligned}
$$

When P and P' are singletons, this definition is equivalent to the acceptance defined for SAAD.

In LSAD, the selection parameter t was estimated by inverting an acceptance function $acc(H, t)$ measuring the acceptance for a chain H. As H is now a set of transitions on *populations*, the acceptance function must also be generalized. Let $acc'(H', t)$ be a measure of the acceptance over a chain where H' is a set of transitions between populations and t the parameter of SELECT. We call it *acceptance function on populations*, and it is defined as follows :

$$
\begin{aligned}
&acc'(H', t) = \mathcal{P}(S \notin P''\ \&\ S' \in P''\ |\ (P \to P') \in H'\\
&\quad \&\ P'' = \text{SELECT}(P, P', t)\ \&\ P''' = \text{SELECT\_REF}(P, P')\\
&\quad \&\ (S, S') \in P \times P'\ \&\ S \in P'''\ \&\ S' \notin P'''\ )
\end{aligned}
$$

Handling a set of transitions between populations would be too complex. It is therefore convenient to transform a set of transitions between populations into a set containing the relevant transitions (between individuals), that is the transitions involved in the conditional part of the acc' function. Formally, given $H' = \{P \to P'\}$, the *set of relevant transitions* (between individuals) is the set

$$
\begin{aligned}
&H = \{S \to S'\ |\ (P \to P') \in H'\\
&\quad \&\ P''' = \text{SELECT\_REF}(P, P')\\
&\quad \&\ (S, S') \in P \times P'\ \&\ S \in P'''\ \&\ S' \notin P'''\ )
\end{aligned}
$$

In Algorithm EAAD, the acceptance is varying from one to (nearly) zero. An acceptance of one implies a selection equivalent to SELECT_REF. An acceptance of zero implies a random selection out of the union of the populations. Therefore, SELECT can be seen as a variable relaxation of the reference selection rule.

We are now in position to present two particular SELECT functions, and their associated ESTIMATE_PARAMETER. In both selections, a relaxation is introduced using the Metropolis criterion.

3.2 2-Metropolis-Tournament

In a 2-tournament, pairs of individuals are formed and for each pair, the best individual of the two is selected. We have developed a relaxed 2-tournament using the Metropolis criterion. The idea is use the Metropolis criterion on each pair to elect the selected individuals. When the winner is elected using a non-deterministic criterion, fairness imposes that each individual enters in a constant number of trials. Therefore, we have decided to make pairs without replacement. Moreover, since the Metropolis criterion is asymmetrical (favors new solutions), we have chosen to make asymmetrical pairs: the first individual always comes from the current population and the second from its offspring. In this case, unless a lazy approach is used, it is useful to impose that the populations P and P' have the same size n.

function $P'' = \text{SELECT}\ (P,\ P',\ t)$
Pre: $\#P = \#P'$
begin
1 $n := \text{SIZE\_OF}\ (P)$
2 $P'' :=$ **array:** $[1..n] \rightarrow$ Individual
3 $P' := \text{PERMUTE}\ (P')$
4 **for each** $i \in [1..n]$:
5 $P''[i] := \text{METROPOLIS}\ (P[i],\ P'[i],\ t)$
end

Algorithm 5: 2-Metropolis-tournament

A 2-Metropolis-tournament is implemented in Alg. 5. In line 3, P' is rearranged in a random order so that $(P[i],\ P'[i])$ form random asymmetrical pairs without replacement. In line 5, the Metropolis criterion (see Alg. 2) is used on each pair to elect a winner which is added to P''.

This selection rule could be used in any EA extended with temperature or acceptance schedule.

3.3 2MT: EAAD with 2-M.-tournament

One could show that in 2-Metropolis-tournament, $acc'(H', t) = acc(H, t)$, where H is the set of relevant transitions from H' (acc is the acceptance function defined in Section 2). Intuitively, individuals of P' are selected independently of each other (this is also true for P). Therefore, the transitions between individuals are also accepted independently.

In the context of 2MT, the set of relevant transitions becomes $H = \{S \rightarrow S' \mid P \rightarrow P' \in H'\ \&\ (S, S') \in P \times P' \&\ \text{ENERGY}(S') > \text{ENERGY}(S')\}$. Therefore, the parameter t can be estimated using ESTIMATE_PARAMETER as implemented in SAAD.

In practice, a sample (of size L) of H can be used in place of H (of size $L \cdot n$). Such a sample can be formed easily by random non-improving transitions from P to P'. This function is denoted SAMPLE_TRANSITIONS.

The proposed EAAD algorithm must now be accommodated to maintain this set of transitions.

3b $H = \emptyset$
6.1 $t := \text{ESTIMATE\_PARAMETER}(H,\ \chi,\ \text{t})$
6.2 $H = \emptyset$
9.1 $H = H \cup \text{SAMPLE\_TRANSITIONS}(P, P')$
9.2 $P := \text{SELECT}\ (P,\ P',\ t)$

One could easily show that 2MT has the same temporal complexity than SAAD, that is $O(s_{1/2} \cdot v \cdot \log(v))$, where v is the average size of a neighborhood.

3.4 Relaxed Truncation

In this section, we design a new selection rule based on a relaxed truncation using the Metropolis criterion. What is needed is a relaxed sorting algorithm; the quality of sorting being subject to a parameter t. Different schemes could be used: in a first one, the standard key comparator is replaced by a stochastic one; in another one, the keys receive a stochastic amount of perturbation. With the first scheme, the acceptance function depends on the chosen sorting algorithm. Therefore, the second scheme is preferred.

The Metropolis criterion can be viewed as a way to sort two solutions. Imagine the situation where S and S' are solutions, x and x' their respective energy, and that S is better than S' (i.e. $\Delta = x' - x > 0$). In the Metropolis algorithm (Alg. 2), we see that S' wins (becomes first) when

$$r < \exp(-\Delta/t) \quad \Leftrightarrow \quad x' < x - t \cdot \ln(r)$$

where r is a random value with a uniform distribution over $[0..1]$. When $t = 0$, a non improving transition can never be accepted. When t increases, an increasing value is added to x and therefore the probability that S' wins increases. The Metropolis criterion is asymmetrical: a penalty is added to x only.

This scheme is extended to populations in Alg. 6. Every individual of P receives a penalty of the form $-t\ln(r)$ (with a different r for each individual). Individuals of P' have no penalty. The mapping v contains the energy plus penalty of every individual. The individuals of $P \cup P'$ are sorted according to this mapping. The set P'' is composed of the first individuals of the sorted union such that P'' and P have the same size.

function $P'' = \text{SELECT}\ (P,\ P',\ t)$
begin
$v :=$ **map:** Individual $\rightarrow$ real
for each $p \in P$:
$v[p] := \text{ENERGY}(p) - t\ln(\text{RANDOM}(0, 1))$
for each $p \in P'$: $v[p] := \text{ENERGY}(p)$
$P'' := P \cup P'$
$\text{SORT}\ (P'',\ v)$
$\text{TRUNCATE}(P'',\ \text{SIZE\_OF}(P))$
end

Algorithm 6: Relaxed truncation

The relaxed truncation selection rule can be used in any EA extended with temperature or acceptance schedule.

3.5 RT: EAAD with Relaxed Truncation

Relaxed truncation accepts individuals using a global approach. Therefore, transitions between individuals are not accepted independently. Hence the relation $acc' = acc$ does not hold here. However, strong experimental evidences show that

$$acc'(H', t) = \frac{n}{n+m} \cdot acc(H, t)^k \quad (8)$$

where k is a constant and H is the set of relevant transitions from H' (with SELECT_REF implemented by Truncation). The term $n/(n+m)$ is the upper bound of acceptance for SELECT (when $t = \infty$). The constant k appeared to be independent of the statistical distribution of the transitions and can be easily computed by simulation. In practice, $k = 0.5$ when the population is large $((n, m) = (7, 50))$ and decreasing slowly toward 1 for smaller populations.

To complete the algorithm, a sample of the set of relevant transitions (e.g. SELECT(P,P',0) in place of SELECT_REF) should also be computed here. This sampling can be achieved as in 2MT. The resulting complexity of SELECT and small SAMPLE_TRANSITIONS is $O((n + m) \cdot \log(n + m))$. Moreover, since ESTIMATE_PARAMETER inverts $acc(H, t)$ and not $acc'(H', t)$, its argument must also be adapted using Eq. 8.

6.1.1 $\chi' = (\chi/(n/n + m)) * *(1/k)$
6.1.2 $t :=$ ESTIMATE_PARAMETER(H, χ', t)

As the ratio n/m is a constant (generally fixed to 1/7), one could easily show that the temporal complexity of RT is $O(s_{1/2} \cdot v \cdot \log(v) \cdot \log(n))$.

4 Experimental Results

The aim of this section is to compare experimentally the proposed algorithms to relevant EA and SA algorithms. For space reasons, only the most relevant experiments are reported. Our analysis is however based on the whole set of experiments.

SAAD is first experimentally compared to classical SA algorithms (SAGEO, SAPOLY and SAEF). For each of these algorithms every parameter is set according to their respective authors recommendations. When a range is proposed, the best values in that range are used. Benchmarks are performed on three TSP instances from the TSPLIB (Reinelt, 1991) (berlin52, ch130 and a280) and on F6. F6 (Davis, 1991) is a moderately multimodal function of low dimensionality ($k = 3$). Table 1 summarizes these experiments. s is the total number of iterations, ϵ is the relative error of the energy of the final solution (compared to the known optimal energy), $s_{1/2}$ is the number of iterations giving a success rate equivalent to 50%. Each line is the result of at least 100 runs.

Problem	Algorithm	Parameters	Results
F6	SAEF		$n_{1/2} = 565\ 296$
F6	SAAD		$n_{1/2} = 429\ 286$
berlin52	SAGEO		$n_{1/2} = 368\ 211$
berlin52	SAPOLY		$n_{1/2} = 223\ 949$
berlin52	SAEF		$n_{1/2} = 150\ 156$
berlin52	SAAD		$n_{1/2} = 163\ 838$
ch130	SAEF	$s = 900\ 000$	$\epsilon = 2.83\%$
ch130	SAAD	$s = 900\ 000$	$\epsilon = 2.78\%$
a280	SAEF	$s = 5\ 250\ 000$	$\epsilon = 3.42\%$
a280	SAAD	$s = 5\ 250\ 000$	$\epsilon = 3.20\%$

Table 1: Comparaison of SA algorithms

	F6		F9		
Algorithms	p	Err.	$\bar{e}$	Std e	Err.
SAAD	18%	±2.4%	265	79	±31
2MT (20,20)	90%	±5.9%	58	13	±5
RT (7,50)	98%	±2.7%	97	22	±9
SHC	6%	±4.7%	260	34	±13
2T (20,20)	26%	±8.6%	71	13	±5
T (7,50)	15%	±7.0%	88	17	±7
ES (7,50)	23%	±8.2%	32	11	±4
ES (30,200)	50%	±9.8%	183	76	±30

Table 2: Results for F6 ($k = 3$) and F9 ($k = 30$)

This study concludes that SAAD and SAEF outperform SAGEO and SAPOLY. SAAD and SAEF exhibit similar performances on the TSP instances with a slight advantage for SAEF on the largest instance. SAAD shows better performances on F6.

Since selection is independent from the problem and from the mutation / recombination operators (Bäck, 1994), our algorithms (SAAD, 2MT and RT) are compared with other EA, identical in every aspects, but using standard selection rules. These algorithms do not use an acceptance schedule nor another form of adaptative selection. They are denoted by SHC (classical Stochastic Hill Climbing), 2T (EA with select = 2-Tournament), T (EA with select = Truncation as in ES-(n+m)). It is also interesting to compare the proposed algorithms to Evolution Strategies (ES) since they use a different (but complementary) approach.

These seven algorithms are compared on two function optimization problems (F6 and F9). F9 is the well-known problem due to Rastrigin generalized as in (Yao and Liu, 1997). It has a high dimensionality ($k = 30$) and is highly multimodal; it is considered difficult for most optimization methods.

For ES, every parameter or choice is made in conformance with the recommendations found in (Bäck, 1996): k standard deviations are used; the solutions are recombined either using a discrete or a panmictic discrete operator (depending on what make most sense for each problem); the standard deviations are recombined using a panmictic intermediate operator; the selection is either ES-(n+m) or ES-(n,m) (whichever gives the best results). For the other algorithms: a log-uniform mutation is used (a value $s \cdot 10^r$ is added to each coordinate, where s is a random sign +1 or -1 and r is a random uniformly distributed real value over [1,-4]); the recombination of the solutions are the same as for ES. For SAad, 2MT and RT, $s_{1/2} = 10$, $L = 1000$ and $stop = 10$. This lead to a total of 144000 generated (and evaluated) individuals. In order to have a fair comparison, all the algorithms are terminated when this number of generated individuals is reached.

All the compared algorithms have been implemented in Java. Source code is available upon request to the first author.

The optimal energies of F6 and F9 is 0. For F6, we measure the proportion p of 100 (independent) runs of the algorithms that lead to the optimal solution (i.e. with a maximal error of 1e-4). For F9, the optimal solution were never reached during the experiments, therefore, we measure the mean energy $\bar{e}$ of the best individual of the final population on 25 runs. Confidence intervals of 95% for p and $\bar{e}$ are also computed. The results are summarized in Table 2.

On F6, 2MT and RT are the best performers by far with ES being in third place. On F9, ES is best and 2MT is second.

On these experiments, the population based algorithms (2MT, RT, 2T, T and ES) show generally better performance than the corresponding solution based ones (SAad and SHC). The algorithms based on an adaptative selection (SAad, 2MT and RT) or on an adaptative mutation (ES) perform generally better than their non adaptative counterparts (SHC, 2T and T). The adaptative selections perform particularly well on F6 and the adaptative mutations on F9. Both approaches are complementary and could be combined.

5 Conclusion

In this paper we designed and experimented three new local search and evolutionary algorithms (SAad, 2MT and RT). They are based on acceptance schedule, an original a schedule for the selective pressure. The successive values of a selection parameter are computed in order to achieve an acceptance schedule. This was impossible with traditional LS and EA algorithms. We thus demonstrate a possible way of merging SA and EA technologies.

Our notion of *acceptance* is a measure of compromise between exploitation and exploration. It takes into account the selection and the generation of neighbors, and is appropriate for both LS and EA.

Adaptable acceptance has been introduced in EA through two new selection rules, introducing the Metropolis criterion on population.

The temporal complexity of the algorithms has been analyzed. They can be computed *a priori*, hence the execution time can be predicted.

Experiments show that the developed algorithms are more performant than standard SA and EA algorithms, and that SAad is as efficient as the best SA algorithm while 2MT and RT are complementary to Evolution Strategies.

This research aims at developing adaptability in LS and in EA. Acceptance driven algorithms should be seen as a possible way to introduce adaptability. Adaptative mutation, such as in (Bäck, 1996), is a complementary approach. Further work includes the combination of acceptance schedule with adaptative mutation, and a characterization of classes of problems where acceptance schedule can be fruitful.

References

E. Aarts and J. Lenstra, editors. *Local Search in Combinatorial Optimization.* John Wiley & Sons, 1997.

T. Bäck. Selective pressure in evolutionary algorithms: A characterization of selection mechanisms. In *Proc. of the First IEEE Conf. on Evolutionary Computation*, pages 57–62. IEEE Press, 1994.

T. Bäck. *Evolutionary Algorithms in Theory and Practice: evolution strategies, evolutionary programming, genetic algorithms.* Oxford University Press, 1996.

L. Davis, editor. *Handbook of Genetic Algorithms*, 1991. Van Nostrand Reinhold.

D. Fogel. *System identification through simulated evolution: a machine learning approach to modeling.* Ginn Press, 1992.

D. Goldberg. A note on boltzmann tournament selection for genetic algorithms and population-oriented simulated annealing. *Complex Systems*, 4:445–460, 1990.

J. Holland. *Adaptation in Natural and Artificial Systems.* Ann Arbor: The University of Michigan Press, 1975.

S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1955.

G. Reinelt. Tsplib - a traveling salesman problem library. *ORSA Journal on Computing*, 3(4):376–384, 1991.

P. van Laarhoven. *Theoretical and Computational aspects of Simulated Annealing.* PhD thesis, Erasmus University Rotterdam, 1988.

P. van Laarhoven and E. Aarts. *Simulated Annealing: Theory and Applications.* D. Reidel Pub. Company, 1987.

X. Yao and Y. Liu. Fast evolution strategies. *Control and Cybernetics*, 26(3):467–496, 1997.