Université de Montréal

Estimation of Noisy Cost Functions by Conventional and Adjusted Simulated Annealing Techniques

par
Laila Abodinar

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Mémoire présenté à la Faculté des études supérieures
en vue de l'obtention du grade de Maître ès sciences (M.Sc.)
en informatique

Décembre, 2017

ABSTRACT

The Simulated Annealing (SA) algorithm is extensively used in the optimization community for solving various kinds of problems, discrete and continuous. This thesis aims to analyze SA in both deterministic and stochastic environments for discrete problems. Precise objectives are to classify key problems, offer suggestions and recommendations to be undertaken by using SA and Simulated Annealing Under Noise (SAUN).

More specifically, problems appear in optimization due to the existence of noise when evaluating the objective function, and how to control this noise. We propose a method, called Noisy Simulated Annealing (NSA), based on the Metropolis-Hasting algorithm modification presented by Ceperlay and Dewing, that outperforms analogous SA techniques, delivering similar numerical solutions, at a reduced cost. We consider the main approaches in the SA setting that handle noise in order to extract their distinctive attributes and make the comparison more relevant. We next assess the numerical performance of the approach on traveling salesman problem instances. The outcomes of our tests show a clear advantage for NSA when solving different problems to get high-quality solutions in presence of noise.

Keywords: optimization, simulated annealing, noisy simulated annealing, random noise, convergence speed, acceptance functions, discrete optimization.

RÉSUMÉ

L'algorithme de recuit simulé est largement utilisé dans la communauté d'optimisation pour résoudre divers types de problèmes, discrets et continus. L'objectif de cette thèse est d'analyser le recuit simulé dans des environnements déterministes et stochastiques pour des problèmes discrets. Les objectifs précis sont de classer des problèmes clés, d'offrir des suggestions et des recommandations à suivre en utilisant l'algorithme de recuit simulé et de recuit simulé sous bruit.

Plus spécifiquement, des problèmes apparaissent en optimisation en présence de bruit, et sur la manière de le contrôler. Nous proposons la méthode de recuit simulé bruité (NSA: Noisy Simulated Annealing), basée sur la modification de l'algorithme de Metropolis-Hastings présentée par Ceperlay and Dewing, qui surpasse les techniques de recuit simulé analogues, délivrant des solutions numériques similaires, à coût réduit. Nous considérons les principales approches qui traitent le bruit dans le cadre du recuit simulé afin d'en extraire leurs attributs distinctifs et de produire une comparaison plus pertinente. Nous évaluons ensuite les performances numériques de l'approche sur des instances du problème du voyageur de commerce. Les résultats obtenus montrent un clair avantage pour le recuit simulé bruité, en présence de bruit.

Mots-clés: optimisation, recuit simulé, recuit simulé bruité, bruit aléatoire, fonctions d'acceptation, vitesse de convergence, optimisation discrète.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CD Ceperly and Dewing

GP Gutjahr and Pflug

i.i.d. Independent and identically distributed

MH Metropolis-Hastings

NSA Noisy simulated annealing

SA Simulated Annealing

SAA Sample Average Approximation

SANE Simulated Annealing in Noisy Environment

SAUN Simulated Annealing Under Noise

TSP Traveling salesman problem

TSPLIB Traveling salesman problem library

1%opt 1%–optimal

NOTATION

T_i Initial temperature

T_f Final temperature

E_i Cost (energy) of solution i

ΔE Cost (energy) difference

$\mathbb{E}$ Expectation operator

σ Standard deviation

σ^2 Variance

σ_i^2 Initial variance

$N(0, \sigma^2)$ Normal distribution with mean zero and variance σ^2

E The Space of the problem

N Tour size, i.e. the number of cities to visit

r_k The run-length

x^* Optimal solution

$d(i, j)$ Distance between city i and city j

P_{ij} Acceptance probability from state i to state j

(dedicace) Cras semper lorem nec pede.

This project is dedicated to the most important persons in my life. Parents, brothers and sisters, all of them emphasize the significance of education in my life. Thank you for everything. No word is enough to express my thanks.

ACKNOWLEDGMENTS

I would like to express my deepest gratitude, and very special thanks to my great and smart supervisor Professor, Fabian Bastin for his support, guidance, efforts, motivation, and great patience. With his wisdom, experience, invaluable remarks and ideas, and timely suggestions at every stage of my research, everything becomes solvable. Thanks also to you for many interesting discussions. Certainly, the words are not enough to describe your help, respect, and kindness.

I am thankful to Libyan Ministry of Higher Education for providing the funding for my study, and many thanks to Canadian Bureau for International Education (CBIE) for their collaboration. Special thanks to Céline Bégin for her advice. Finally, I am also grateful to my professors, teachers, colleagues, and friends on this support.

CHAPTER 1

INTRODUCTION

1.1 Background

Optimization problems are notoriously present in a myriad of disciplines and considered in industry, government, and academia (see for instance [63], p. 13); yet many of these problems cannot be solved in polynomial running time. Heuristic methods will provide improved practical solutions for these kinds of problems. Based on many research studies, heuristic and meta-heuristic methods are broadly used for solving numerous problems [2]. A stochastic optimization algorithm is a simulation optimization method that operates stochastic steps in order to discover global or local minimizer for the problem. Due to the inherent uncertainty, the outcomes can differ between several executions, relying on the injected randomness that can be produced from a particular simulation. On the opposite, a deterministic algorithm always produces the same outcomes, provided we start the process with the same initial conditions or sequence of steps (Aguiar e Oliveira Jr et al. [4], chapter 1, p. 6).

One of the most salient concepts that this thesis deals with is annealing, which is ubiquitous in varied areas and it is used for combinatorial and continuous global optimization techniques (see for instance [47] and [63], chapter 8, p. 1). Annealing can be defined according to Merriam-Webster Dictionary as the process used to "make metal or glass soft by heating and then cooling it slowly". Combinatorial optimization, statistical physics, and applied physics are different fields, but it is notable that the Simulated Annealing (SA) algorithm originally relies on an analogy between them [2]. SA procedure, described as "the most exciting algorithmic development of the decade" by Fabian [24] in 1997, uses the annealing technique, and it is crucial to put our focus on it.

In the simulation optimization area, many developments towards other metaheuristics have been considered since the introduction of SA, such as Genetic Algorithm (GA), Tabu Search, and Scatter Search, leading to an impressive number of publications [7].

Nevertheless, many applications take place in noisy environments, while these techniques usually ignore this aspect. The second important concept in this thesis is therefore noise. According to the Cambridge Dictionary, noise means "irrelevant or meaningless data or output occurring along with desired information". Even for optimization problems where SA delivers good results, a few drawbacks appear for noisy environment as it produces biased results when applied [15]. It is important to know how to adapt SA to deal with noisy problems in order to estimate a solution, which is sensitive to the noise for different reasons, therefore, there is a need to find a heuristic technique that is able to cope with this noise. We consider Simulated Annealing Under Noise (SAUN) algorithms that may overcome this drawback as they aim to optimize objective functions where each feasible solution, the corresponding cost follows a random distribution based on the uncertainty of the inputs. We then propose a novel approach, called Noisy Simulated Algorithm (NSA) to improve the efficiency of SAUN.

1.1.1 Applications of SA

Vidal [66] highlights various applications of SA algorithm, in mathematics, e.g. graph problems, in physics, e.g. finding the ground state of spin glasses, in engineering such as very-large-scale integration (VLSI) design, and more generally operations research, e.g. combinatorial optimization and neural computing optimization. Ingber [35] mentions some other applications, for instance, the traveling salesman problem (TSP), circuit design, data analysis, imaging, neural networks, biology, physics, geophysics, finance, and military, while Duque-Antón [22] mentions the channel assignment problem occuring in the design of cellular radio systems. Other lists of applications can also be found in [1, 2, 41].

1.1.2 TSP

TSP is often used as a benchmark for SA [35], so we will focus on this application. According to Rego and Glover [54], from 1993 to mid 2001, more than 150 papers dedicated to the TSP are listed by the web databases of INFORMS and Decision Sci-

ences. Moreover, Kirkpatrick et al. [40] expose the possibility to tackle a number of problems of scheduling and design, for instance, to anticipate the expected cost of the salesman's optimal route. Johnson and McGeoch [36] add that TSP approach can be applied to other applications as VLSI chip design, X-ray crystallography, etc. Punnen [53] presents applications such as machine scheduling problem, cellular manufacturing, arc routing, frequency assignment, matrices structuring, printed circuit boards drilling, gas turbine engines overhauling, order-picking problem in warehouses, computer wiring, data clustering, archeology serration, vehicle routing, mask plotting in PCB production, robot control, etc.

There exist several software tools that use SA as a heuristic method for solving TSP as listed by Lodi and Punnen [45], for instance "parSA-Lib", a general-purpose C++ library for applying simulated annealing algorithms in parallel, "RA-TSP", solving "a variant of the ATSP called Arc Replenishment Traveling Salesman Problem", and the Mathematica package "Operations-Research- 2.0".

1.2 Thesis Motivation

The main motivation behind our research is that simulated annealing remains a popular optimization method in many industries when the objective function is evaluated by simulation. SA is often considered in order to interpret, understand, and optimize complex systems, but there exist limited investigations that consider noisy information in applications [63], chapter 8, p. 7. For this reason, we suggest using SAUN to deal with such problems in order to mimic the reality and to get a better understanding for some combinatorial problems such as the TSP [15]. SA and SAUN are the most significant heuristic algorithms that will be used in our research work. Our main goal is to understand them more precisely, their behavior and their properties, and evaluate the solutions quality when the noise is an undeniable part of the objective function, in the context of the TSP.

1.3 Thesis Statement and Objectives

To date, very limited attention has been given to the accuracy of the function evaluation. The standard simulated annealing rule is to accept a point as soon as a better objective value function is obtained, otherwise, the point is accepted with a probability that decreases with the iteration index. As a result, the optimal solution and the optimal value can be biased, especially as the accuracy is often limited in order to perform more iterations within a time budget.

The project first aims to build an experimental framework allowing to numerically explore new strategies to handle noise inside the simulated annealing framework and to compare the quality of obtained solutions. We consider standard TSPs to empirically compare the various approaches that are currently developed and analyzed in parallel projects. This study aimed to determine whether SAUN is able to get optimal convergence in practice and to elucidate its behaviour and its efficiency on noisy TSPs.

Our main contribution is the development of a new SA variant that can work more proficiently on noisy problems, and significantly outperforms other SAUN methods in terms of the computation cost and quality of solution. The technique, called noisy simulated annealing (NSA), controls the noise by using the modification initially proposed by Ceperley and Dewing [16] for the Metropolis-Hastings acceptance criterion, in a fixed temperature and random error setting. We adapt their approach by controlling the noise level at the initial temperature and slowly reduce it along with the temperature decrease.

1.4 Thesis Organization

Chapter 2 lays facts on simulated annealing algorithm and its ingredients. Chapter 3 gives background and more information about simulated annealing in presence of noise. Chapter 4 contains detailed information on TSP. Chapters 5 and 6 investigate the experimental results including numerical outcomes and figures based on different aspects. Chapter 7 concludes the research work. It recaps the thesis and presents a summary of limitations that we have faced and avenues for future work.

CHAPTER 2

SIMULATED ANNEALING

2.1 Background

Simulated annealing has been studied and considered by many researchers (see for instance [1–3, 12, 17, 19, 34, 35, 47, 49, 57, 58, 68, 69, 71], [63], chapter 8, p. 1, [28], chapter 3, p. 10, Schneider and Kirkpatrick [61], chapter 11, p. 78). Simulated Annealing was conceptualized by Kirkpatrick et al. [40] in 1983 and by Černỳ [17] in 1985. Many researchers called SA with various aliases, such as "Monte Carlo annealing", "statistical cooling probabilistic hill climbing", "stochastic relaxation" or "probabilistic exchange algorithm". According to van Laarhoven and Aarts [41], "simulated annealing algorithm is based on the analogy between the simulation of the annealing of solids and the problem of solving large combinatorial optimization problems. For this reason, the algorithm is known as simulated annealing". Many authors report that SA algorithm is a well-adjusted version of iterative methods, and is a heuristic approach to solve optimization problems [41], and [51], chapter 4, p. 187.

The basic philosophy of SA is to mimic the annealing process in the metalwork, which briefly involves two steps. First, solid metal is put in a heat bath and the temperature is raised till the solid melts [60], and "the atoms gain enough energy to break the chemical bond and become free to move" [59]. Second, the metal is cooled lingeringly and slowly until its particles are reordered in the "ground state of solid". Consequently, the metal is now differently characterized since the process helps to reattain appropriate crystal structure with an idealistic grid, with minimal energy. SA algorithm acts in a similar way to find out an optimization problem solution. Initially, it begins with arbitrary configuration, and at each single point, it randomly chooses the next configuration from neighbor space configurations with a small distortion. The neighbor is always accepted if the objective function value is decreased, and with some random rate, decreasing with the process iterations, if the objective function value is increased, till it reachs the global

optimal configuration when the temperature is frozen and obtaining optimum solution is akin to getting the least energy state as the operation ends [1–3, 12, 26].

According Saıt and Youssef [60], the simplicity of the representation is significant in order to obtain reasonable performance as the algorithm might need a large number of iterations. Three main requirements are needed to use SA. Firstly, the state space must be concisely and clearly interpreted and the cost function that will be determined to get a solution should not be complicated to calculate. Second, it requires a mechanism to transform a solution to another one during the search operation to find the next move. This step has two main ingredients, a neighborhood search and an acceptance mechanism based on the cost difference between the current and the candidate solutions. More precisely, given a current solution, the neighborhood search can select any solution in its vicinity as the next solution with some positive transition probability, defining a Markov chain, and any solution in the system can be produced in a finite number of moves, meaning that the Markov chain is irreducible. This chain is also aperiodic as given any pair of solutions, the possible number of moves such that the probability to attain the second solution given the first one is positive, define a set of naturals with no common divisor other than 1. Finally, the efficiency of SA relies on a suitable choice of cooling schedule.

As illustrated in Figure 2.1, reproduced from [51], the algorithm starts from some initial solution, associated to a high temperature T_0. The algorithm then iteratively generates candidate solutions, while the temperature is decreased. At each step, the candidate solution will be accepted as the new solution if the objective cost is decreased, but will be rejected with some probability if the cost is increased. Nevertheless, the opportunity of admitting a solution with higher cost will decline as the temperature T reduces, and ultimately, the probability to accept a solution with higher energy converges to 0 as T goes to 0 [9]. By applying this strategy, the algorithm is allowed to gradually target a space hopefully close to the optimal solution, and the sequence of solutions can be seen as a stochastic process, with transition probabilities evolving over the iterations. The SA method can be summarized in algorithm 1 (see for instance [15, 26]). We give the main steps below, and will give implementation details in the following sections.

1. Select an initial solution x_c and temperature T_0. Set the iteration index k to 0.

2. Repeat r_k times the following.

   (a) Select a candidate solution from a neighborhood of x_c, $\mathcal{N}(x_c)$, and compute the difference of energies $\Delta E_{nc} = E(x_n) - E(x_c)$.

   (b) Accept the candidate solution x_n with a probability P_{nc}, increasing with $-\Delta E_{nc}$: $x_c \leftarrow x_n$.

3. Set $T_{k+1} < T_k$.

4. Stop if some termination criterion is met, otherwise set $k \leftarrow k+1$ and return to step 2.

When r_k is greater than 1, the inner loop is usually executed until equilibrium is approached sufficient closely for the current temperature T_k. The SA algorithm is then said to be homogeneous. In inhomogeneous SA, r_k is equal to 1 for all k, and the temperature is decreased in a lower rate, often very slowly [66].

Algorithm 1 Simulated Annealing algorithm

```
Generate initial solution x_0, and select the initial temperature T_i.
Set x_c = x_0, T_0 = T_i, k = 0.
repeat
    repeat
        Generate a candidate solution x_n ∈ N(x_c)
        Set ΔE_nc = E(x_n) − E(x_c)
        Draw u from U ~ U(0,1)
        if u < P_nc then
            accept new solution: set x_c ← x_n
        end if
    until time to reduce temperature
    Set T_{k+1} = h(T_k).
    Set k ← k + 1.
until termination condition is met
```

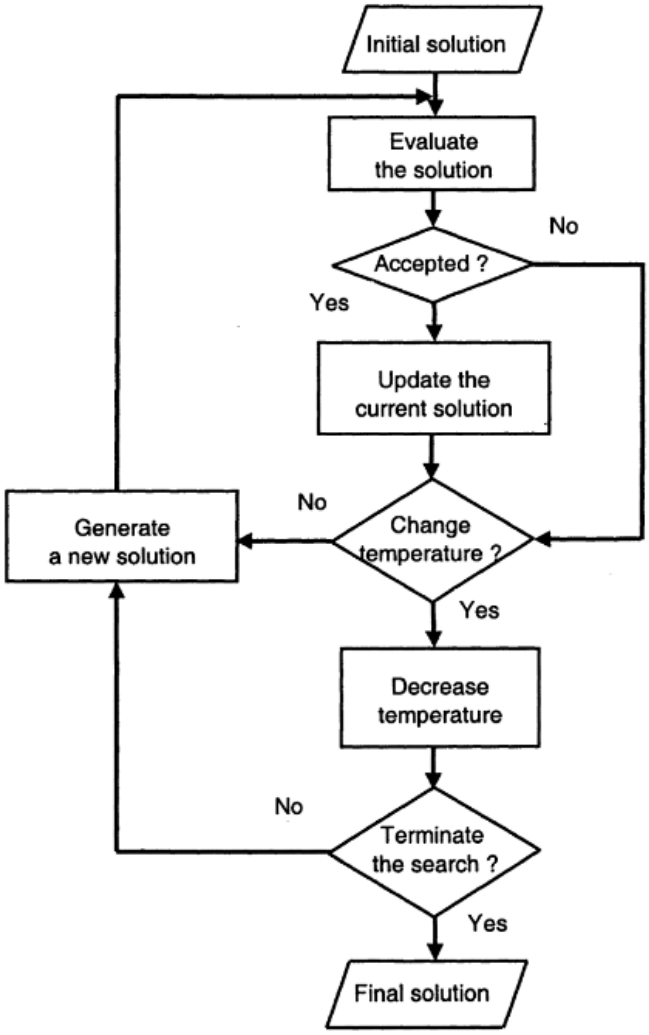


Figure 2.1: SA algorithm

Decisions	
Generic(Cooling Scheme)	Problem Specific
T_0 (initial temperature)	x_0 (initial solution)
r_k (number of iterations)	neighbor generation
T_k (temperature function)	evaluation of ΔE_{ij}
Stopping criterion	

Table 2.I: SA implementation choices

2.2 Simulated Annealing Implementation

It is necessary to set some parameters of the algorithm prior its execution, as illustrated in Table 2.I, due to (Vidal [66], p. 8), and to properly represent the problem to optimize. According to Eglese [23], Fouskakis and Draper [26], Ledesma et al. [43], for instance, each possible state of the system has to correspond to a feasible solution of the optimization problem. The energy level E_i of a state i expresses the cost of the objective function. We detail the specific implementation choices in the following.

2.2.1 Acceptance Function

Several acceptance strategies exist [2, 41], based on the difference between E_j and E_i. According to Anily and Federgruen [9], there exist some conditions favoring the discovery of good solutions. For a certain number of iterations, the algorithm should confer any uphill or downhill to happen with positive probability. According to Henderson et al. [34], "the acceptance probability function must be bounded and asymptotically monotone, with limit zero for hill-climbing solution transitions". Ideally, we should have that the probability to produce a non-globally optimal solution should be asymptotically equal to zero. In practice, the algorithm can however be trapped in a local minimum, so we want that the probability to escape from such a solution does not go to zero too fast. In other words, the probability to accept an uphill move should slowly decrease to 0, and the algorithm should coin the solution to a local minimal, hopefully a global one. Even if the probability for an uphill move is not equal to 0 but small during the final iterations, the probability to accept two consecutive uphill moves is then close to 0, so that

in practice the algorithm can oscillate around the found solution, but not escape from it. Various researchers however state that the convergence of the SA still depends on the initial solution, and the algorithm does not always deliver a globally optimal solution.

2.2.1.1 Metropolis-Hastings Criterion

The most popular acceptance technique is the Metropolis-Hastings (MH) criterion, defined by

$$P_{ij} = \begin{cases} 1 & \text{if } \Delta E_{ij} \leq 0 \\ e^{\frac{-\Delta E_{ij}}{K_B T}} & \text{otherwise,} \end{cases}$$

where K_B is a physical constant called Boltzmann constant. For a high temperature, nearly every move is accepted, but for a low temperature, the probability to accept a state of higher energy is close to 0. Without loss of generality, we can set $K_B = 1$, by scaling the temperature, leading to the acceptance probability

$$P_{ij} = \min\left\{1, e^{-\Delta E_{ij}/T}\right\}. \tag{2.1}$$

2.2.1.2 Glauber's Acceptance Criterion

Other criteria can also be used while ensuring convergence, but at slower rate [1]. It is especially possible to use the acceptance criterion proposed by Glauber [29]

$$P_{ij} = \frac{1}{1 + e^{\Delta E_{ij}/T}} \tag{2.2}$$

Three situations can be considered. When the difference between the current and candidate solutions is equal to 0, then $P_{ij} = \frac{1}{2}$. If $\Delta E_{ij} < 0$, i.e. we consider a downhill move, $P_{ij} > \frac{1}{2}$, while for a uphill move ($\Delta E_{ij} > 0$), $P_{ij} < \frac{1}{2}$. In addition, the temperature has an important role too. The temperature T plays the expected role as with $T \to \infty$, every move will be accepted with a probability equal to $\frac{1}{2}$, while when $T \to 0$, the acceptance probability tends to 1 for downhill moves, and 0 for uphill moves. We can however observe that the diversification effect is less present than with MH for high temperatures,

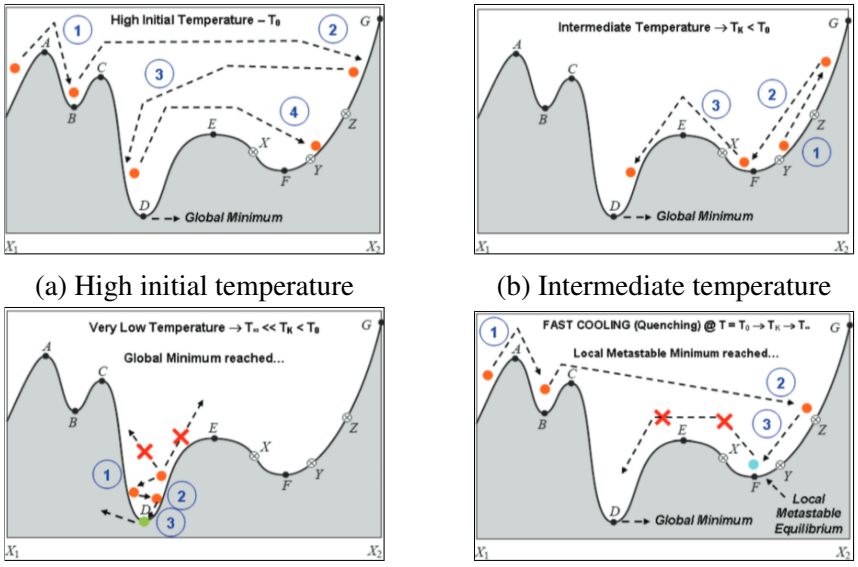
and that downhill moves can be rejected, while they are always accepted with MH.

2.2.2 Cooling Schedule

It is known that the cooling schedule has an impact on the solution quality. According to Pham and Karaboga [51], chapter 4, the main parameters of cooling schedule are: first, the initial temperature T_i; second, the number of function evaluations at each temperature and the temperature update rule; third, the final temperature T_f and a stopping criterion for the search.

2.2.2.1 Initial Temperature T_i

There is no typical criterion to choose the appropriate value factors. If the initial temperature is high, a lot of time will be spent to attain the solution and the cooling process will take a long time. On the contrary, if the initial temperature is very low then the algorithm terminates very rapidly, and the returned solution is usually poor. Figure 2.2 gives an idea of this behavior (see [64], p. 238–239).



(a) High initial temperature

(b) Intermediate temperature

(c) Very low temperature

(d) Fast cooling T_k

Figure 2.2: Temperature setting

Ben-Ameur [11] states that several forms have been suggested to define the initial temperature T_i. In [61], chapter 15, [41], chapter 3, p. 28–38, authors illustrate that some methods are simple but not necessarily efficient, while other methods are efficient but not necessarily simple. The main rule is to explore the entire search space until getting close to good solutions, but this can happen only if the temperature is large enough. Various interesting methods can be found in the litterature [26, 32, 37, 38, 60, 65, 67], and we will present the approaches used in our experiments in chapter 5.

2.2.3 Temperature Update

According to Rosen and Harmonosky [57], the temperature updating method impacts the performance of the SA, and various papers explore the parametrization of SA, for instance [67]. Schneider and Kirkpatrick [61], chapter 15, p. 122, mention various cooling methods:

Geometric Cooling The temperature is updated as

$$T_{k+1} = \alpha T_k$$

where $\alpha \in [0.01, 0.2]$, and k is the iteration index [26, 48, 60, 61]. Rosen and Harmonosky [57] reveal that sometimes, the temperature has to decrease fast during the first iterations, but then the algorithm has to generate "increasingly smaller T_k drops" as SA tends to explore no further positions of the optimal solution area when the temperature is small. Geometric cooling has the interesting property that it allows a large initial temperature T_i.

Linear Cooling The temperature at the iteration k is

$$T_k = a - \alpha k$$

Nourani and Andresen [48] mention that the linear temperature has been extensively applied and was introduced by Kirkpatrick et al. [40]. A variant consists to allow α to be random [40].

Logarithmic Cooling The temperature T_k is obtained as

$$T_k = \frac{a}{b + \log(1+k)}.$$

As observed by Nourani and Andresen [48], the decrease speed is not constant but slows down over the iterations. SA has been shown to almost surely converge to a global minimizer under the logarithmic cooling schedule and some mild conditions (see for instance [20, 47]). Hajek [32] establishes that if $b = 1$, a necessary and sufficient condition on the cooling schedule for the algorithm state to converge in probability to the set of globally minimum cost states is that a is greater than or equal to the depth, suitably defined, as "the deepest local minimum which is not a global minimum state".

2.2.3.1 Final Temperature T_f

T_f describes the temperature used in the last iteration, where a steady state is expected to have been reached. Several approaches can be employed to estimate this value, that will be described in the following chapters.

2.2.4 Stopping Criterion

The main function of stopping criterion is to indicate when the algorithm terminates. (Spall [63], chapter 1, p. 15), emphasizes that, in the context of SA, there is no clear rule to estimate when the algorithm should end. In other words, it is difficult to get a good stopping criterion ensuring optimality. Branke et al. [15] suggest to predetermine the number of iterations or fix the temperature limit if it is practical, but it may take many experiments to find a reasonable value and it is problem dependent. Rutenbar [58] suggests to terminate the algorithm execution when the cost improvement over three successive temperatures, for example, is less than one percent of the optimal solution. According to Eglese [23], this is the most efficient strategy for a general cooling schedule. Saıt and Youssef [60] generalize the criteria by stopping if no improvement has been achieved during the last iterations or is a given time budget has been exhausted or

if some of the SA parameters have reached given thresholds.

2.2.5 SA Repetitions

Several authors suggest to restart the algorithm several times, from different starting points, generating for instance a random initial solution at each repetition [12, 57]. The number of repetitions should however vary with respect to the problem under consideration, as a difficult problem usually needs many restarts while a small problem does not, and could be set by the user as in [57].

2.3 SA Assessment

Due to its simplicity of implementation, SA has been popular for solving various optimization problems, and has benefited from various theoretical analyses [3, 34, 40]. Ingber [35] states one of the interesting features of SA is its ability to "process cost functions possessing quite arbitrary degrees of nonlinearities, discontinuities, and stochasticity", and Ledesma et al. [43] add that there is no need of mathematical paradigm in the solution design. SA uses an iterative method based on local random search, exploration, exploitation, and greed properties, and is seen as effective and robust, as usually a high-quality solution can be obtained, from any selected initial solution [60, 70].

Some authors however express some criticism, as SA is not considered as fast, being "overkill for many of the problems on which it is used" [35]. Vidal [66] highlights that SA is time consuming due to its stochastic approach. In addition, Ingber [35] claims that SA is challenging to be specifically adjusted to the problem under consideration, in addition to producing incorrect outcomes if misused, and Charnes and Wolfe [18] express that SA is mainly based on physical intuition, with not enough mathematical rigor. Finally, according to Saıt and Youssef [60], SA is "blind" as it is no possible to know if the optimal solution has been obtained or not, so the stopping criteria cannot be set as an optimality test. In addition, there is no guarantee to reach optimality, even if almost-sure convergence can be ensured if an infinite number of iterations was allowed [3]; therefore, it is an approximation technique. Aarts et al. [3] add that "Experience shows that

the performance of simulated annealing depends as much on the skill and effort that is applied to the implementation on the algorithm itself; for instance, the choice of an appropriate neighborhood function, of an efficient cooling schedule, and of sophisticated data structures that allow fast manipulations can substantially reduce the error as well as the running time. Thus, in view of this and considering the simple nature of annealing, there lies a challenge in constructing efficient and effective implementations of simulated annealing".

Eglese [23] mention some possible modifications to improve SA efficiency, as the storage of the best found solution during the iterative process, the possibility to sample the neighborhood without replacement, and alternative acceptance probabilities. It is also possible to combine SA with another method, to parallelize, and to implement problem specific modifications.

CHAPTER 3

SIMULATED ANNEALING IN PRESENCE OF NOISE

3.1 Problem Formulation

We consider the stochastic optimization problem

$$\min_{x \in \mathscr{X}} \mathbb{E}_{\omega}(L(x, \omega)), \tag{3.1}$$

where $\mathscr{X}$ is the feasible set and $\omega \in \Omega$ is some random vector capturing the uncertainty in the objective function evaluation. Assuming that (3.1) has a unique solution x^*, we will write

$$x^* = \arg\min_{x \in \mathscr{X}} \mathbb{E}_{\omega}(L(x, \omega)).$$

The expectation can often not be evaluated exactly if it does not have an analytical expression or its evaluation cost is prohibitive if the number of possible realizations of the random variable is finite but large. A popular approach consists to replace (3.1) by its sample average approximation (SAA) obtained by sampling over the random variable

$$\hat{L}_N(x) := \frac{1}{N} \sum_{i=1}^{N} L(x, \omega_i), \tag{3.2}$$

where $\{\omega_i,\ i = 1, \ldots, N\}$ is a i.i.d. Monte Carlo sample (see for instance [13]).

Various issues arise when incorporating noise, as summarized by Spall [63], and illustrated in Figure 3.1. Typically, a noisy perturbation creates many local minima and can offset the global minimum, and considering various realizations of the noise creates a "lack of stationarity in the solution". Finally, increasing the number of Monte Carlo realizations in (3.2) significantly affects the evaluation costs [5], that linearly grow with N. Facing these difficulties, one or our objectives will be to take advantage of the noise in the SA framework instead of trying to remove it.

Adding noise can make the search process more powerful and flexible. It is useful

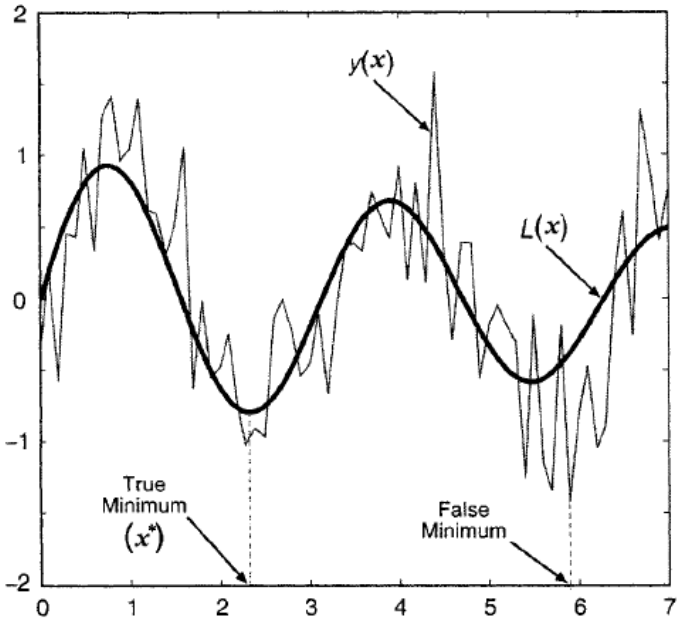


Figure 3.1: Example of function $L(x)$ with minimum x^* along with a perturbated function $y(x)$, producing a false minimum

to mimic the real-world problems, can help to seek a global optimum solution when the search is stalled near a local solution, which is relevant for speed convergence, and make the algorithm less sensitive for error modeling [61, 63]. In stochastic programming, the SAA problem (3.2) is usually solved using common random numbers [62]. We nevertheless here consider independent random numbers for each objective function evaluation as SA is typically used to tackle problems where the random realizations cannot be kept fixed from one iteration to another one, for instance when the objective function is evaluated through a black-box routine. The use of common random numbers when comparing a pair of solutions would result in a decrease of the variance of the difference of energy, possibly allowing a faster convergence of the SA algorithm. The main conclusions would however remain valid as only the error variance scale is changed.

For simplicity, we will assume the the noise at each possible state is additive, so that we can write

$$L(x,\omega) = V(x) + \varepsilon(x,\omega). \tag{3.3}$$

Moreover, if we assume a white noise, i.e. $\mathbb{E}_\omega(\varepsilon(x,\omega)) = 0$, we can rewrite (3.3) as

$$L(x,\omega) = \overline{L}(x) + \varepsilon(x,\omega),$$

where $\overline{L}(x) = \mathbb{E}_\omega(L(x,\omega))$. We will moreover assume that the errors $\varepsilon(x,\omega)$ are i.i.d. normally distributed, allowing to further simplify the expression of $L(x,\omega)$ as

$$L(x,\omega) = \overline{L}(x) + \varepsilon(\omega), \tag{3.4}$$

where $\varepsilon(\omega) \sim N(0,\sigma^2)$. This assumption is quite common in the SA under noise litterature (see for instance [15]).

3.2 Noise Management in SA

According to Spall [63] (chapter 8, p. 7), there is limited research regarding optimization in presence of noise, especially with respect to the impact of statistical errors in the input of the algorithm on the resulting errors in the output. As noted by Saıt and Youssef [60], in the deterministic case, the best solution discovered during the execution of the SA algorithm should be returned, but in the stochastic case, the final solution is more important as the cost stabilizes in the end. The noise at the current iterate can be reduced by averaging over several independent evaluations, but the associated numerical cost is rapidly prohibitive. Spall notes that the major issue lies in the comparisons of energies. Under the assumption of i.i.d. normally distributed error term, as in (3.4), the energy difference ΔE_{ij} between two states x_i and x_j is also normally distributed with mean 0 and variance $\sigma^2_{\Delta E} = 2\sigma^2$.

Branke et al. [15] observe that convergence issues can arise, and a strong noise can slow down the algorithm. The noise can also bias the objective function, and the algorithm can be trapped in a local minimizer, or even produce final solutions of unacceptable quality as they are themselves biased.

Two main approaches have been proposed: reducing the noise over the iterations while keeping the algorithm untouched, or modifying the acceptance function. We will

present them and elaborate on a novel method to adjust the acceptance probabilities and control the noise.

3.2.1 Noise Reduction

The noise can impact strongly on Metropolis criterion, biasing SA algorithm [15]. In particular, noise can reduce the probability to accept a downhill move, while increasing the probability to accept an uphill move, as illustrated in figure 3.2. There is a need to choose many levels of noise that must be applied to the problem to see the effect and to validate the algorithm because it can not be used as it is. Figure 3.2 visualizes these cases.

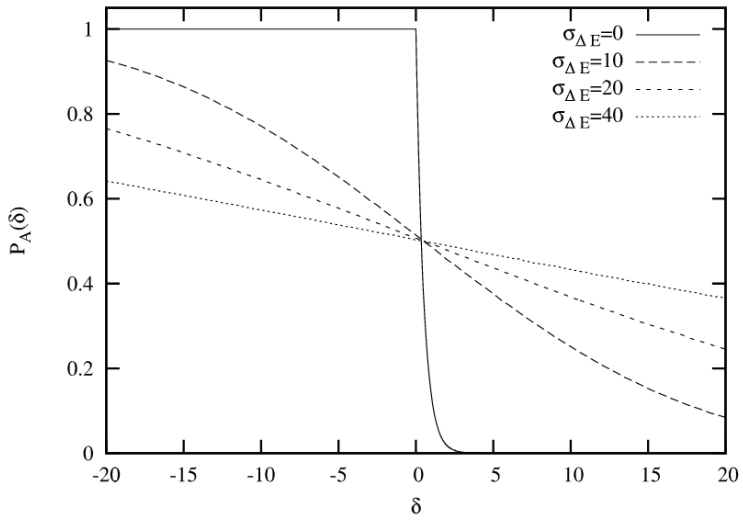


Figure 3.2: MH acceptance probability in presence of noise

The first idea is to reduce the noise as when the temperature is lowering as we try to stabilize the solution. Ultimately, the noise should converge to 0 as the temperature is going to 0, and this strategy has been examined by many authors. We refer the reader for instance to [6, 8, 13, 27, 31, 46, 48, 52].

A simple way to reduce the noise at a given point x is to evaluate several times the objective function and take the average over them. Consider indeed ℓ evaluations. The average noise then follows a normal distribution of mean 0 and variance σ^2/ℓ. This

approach was initially proposed by Gelfand and Mitter [27], who suggest to decrease linearly the standard deviation of a state energy with the temperature:

$$\sigma_k = o(T_k))$$

In terms of sample size, the number of Monte Carlo draws has to increase quadratically with the inverse of the temperature [15].

Gutjahr and Pflug [31] refined the analysis, discussing the convergence of SA under noise. Under the assumption that the standard deviation of the noise is in $O(k^{-\gamma})$, where γ is an arbitrary constant > 1, and the temperature T_k is of order $\Omega(1/\log k)$, they established convergence of SA, but if the variance is unchanged, SA is not capable to reach optimality. They also extended the results to other distributions, that are "more peaked around zero" than the normal distribution.

Bouttier and Gavra [13] argue that this "convergence statement did not give any information about the convergence rate of the algorithm" and investigate the convergence of the method under various temperature cooling schemes, extending Gutjahr and Pflug [31]'s results, but establishing that their approach is optimal in terms on computational efforts if the SA algorithm is kept unchanged.

3.2.2 Acceptance Function Modification

We therefore have to consider algorithmic modifications in order to speed up the method. The most convenient approach is to change the acceptance criterion, that basically relies on the sign of the energy difference: $\Delta E < 0$ or $\Delta E \geq 0$. The basic idea is to replace it by some criteria $\Delta E < \tau$ or $\Delta E \geq \tau$, where τ is some threshold value that may be positive or negative depending on the circumstances. According to Gutjahr [30], various adaptations have been proposed. We will consider two main ideas, proposed by Fink [25] and by Branke et al. [15], who have developed a method called SANE, for Simulated Annealing for Noisy Environments.

3.2.2.1 Stochastic Annealing

In presence of noise, Fink [25] suggests to base the acceptance probability on the observed energy difference, instead of using the MH criterion:

$$P_{ij} = \begin{cases} 1 & \text{if } \Delta E_{ij} \leq 0 \\ 0 & \text{otherwise,} \end{cases}$$

He justifies this approach, called "stochastic annealing", by a graphical analogy with the Glauber acceptance criterion (see section 2.2.1.2), stating that the resulting acceptance probability for the energy difference expectation is then similar. In other words, instead of injecting randomness in the problem when deciding to accept or reject a candidate solution, we exploit the noise already present. Based on this analogy, he also derived a relationship between the number n of Monte Carlo draws and the temperature:

$$\frac{1}{T} = \sqrt{\frac{8n}{\pi\sigma_{\Delta E}^2}}. \tag{3.5}$$

The number of draws has therefore to grow to infinity as the temperature is going to 0, but as stated by Bouttier and Gavra [13], "unfortunately he only provided a few numerical examples to validate his statement and a theoretical proof is still missing".

3.2.2.2 SANE

Branke et al. [15] noted some issues with (3.5). First, when the temperature is high, the corresponding number of draws suggested by the formula can be less than one. Second, since n has to be an integer, the equality can only be satisfied at some specific temperatures. Branke et al. [15] proposed some remedies to these problems in order to permit any temperature level to be used, and presented the Simulated Annealing in Noisy Environments (SANE) algorithm. When the noise is small compared to the temperature, they rely on the Ceperley and Dewing's method, described below, and on the Glauber analogy when the noise is important compared to the temperature. Therefore,

their approach also lacks a formal convergence theory.

3.2.2.3 Ceperley and Dewing's Acceptance Criteria

For a fixed temperature, Ceperley and Dewing [16] (CD) propose to adjust the Metropolis-Hastings criterion as follows:

$$P_{ij} = \begin{cases} 1 & \text{if } \Delta E_{ij} \leq -\frac{1}{2}\sigma^2_{\Delta E_{ij}}/T, \\ e^{-(\Delta E_{ij}/T + \frac{1}{2}\sigma^2_{\Delta E_{ij}}/T^2)} & \text{if } \Delta E_{ij} > -\frac{1}{2}\sigma^2_{\Delta E_{ij}}/T. \end{cases} \tag{3.6}$$

(3.6) can be seen as an generalization of MH in presence of noise, but reduces to it in the deterministic case. In average, less moves are accepted using (3.6) instead of MH, but they prove that the method then converges to the correct equilibrium distribution when the noise is normally distributed with mean 0. They also briefly discuss the situations where the variance is observation-dependent or where the noise follows other distributions than a normal distribution.

Branke et al. [15] dismissed the CD approach as when keeping the noise level fixed, without consideration of the temperature, the acceptance probability quickly goes down to 0, irrespectively of the sign of the energy difference.

3.3 Noisy Simulated Annealing

As previously stated, the method developed by Ceperley and Dewing does not consider a varying temperature, and as such, should not be applied to SA without modifications. Ceperley and Dewing [16] briefly discuss the impact of noise magnitude, and exhibit that the approach does have a clear benefit when the noise is too small, and will face issues when the noise is too large. Therefore, the noise should be adjusted when the temperature is dropping in order to preserve CD method qualities.

A closer examination of (3.6) suggest to maintain the inequality

$$\sigma^2_{\Delta E} \leq \kappa T^{\eta} \tag{3.7}$$

valid for any temperature T, with $\eta \in (1,2]$ and $\kappa = \frac{2\sigma_0^2}{T_0^\eta}$. In other terms, assuming that the noise level, expressed as the variance, at the initial temperature has been well chosen to favor a CD approach, we decrease the variance when the temperature is lowering in order to prevent the acceptance probability associated to a downhill move ($\Delta E < 0$) getting smaller. When $\eta = 2$, we obtain a variance decrease similar to the recommendation given by Gutjahr and Pflug [31], but the approach still works with values of η close 1, leading to a much slower variance reduction. As we will see in the numerical experimentations, the effect on the computing cost is then significant.

CHAPTER 4

TRAVELING SALESMAN PROBLEM

The traveling salesman problem has often been used to evaluate the performance of SA implementations, in particular using instances from the problem collection TSPLIB. We briefly present the problem in this chapter.

4.1 TSP Description

Consider a set of $\mathbb{N}$ cities, and a salesman that must visit each of them once and only once and then return to his home city. The traveling salesman problem (TSP) is to compute the shortest (connected) tour. The distance between two cities i and j is denoted by $d(i,j)$. TSP is called symmetric if $d(i,j) = d(j,i)$ for all i,j. This is in particular the case if the city position is described by a 2-dimensional vector of coordinates, we use the Euclidean distance to compute the distance between two cities. Without loss of generality, we set the index of the home city to 1. A tour can be described by permutation of the cities 2 to $\mathbb{N}$: $\pi = (\pi(2),\ldots,\pi(\mathbb{N}))$. The complete connected tour is then $(1,\pi(2),\ldots,\pi(\mathbb{N}),1)$, and its length is

$$H(\pi) = d(\pi(\mathbb{N}),\pi(1)) + \sum_{i=1}^{\mathbb{N}-1} d(\pi(i),\pi(i+1)),$$

where by convention $\pi(1) = 1$. The solution space can therefore be described as the set $S = \{\text{all permutations } \pi \text{ on } \mathbb{N}-1 \text{ cities}\}$. The dimension of the solution space is therefore $|S| = (\mathbb{N}-1)!$ (see for instance [3]). We can also describe the problem using graph theory [36]. The tour is a Hamiltonian cycle in a graph where every node or city has to be visited once.

The TSP has received a lot of attention (see for instance [10, 14, 33, 36, 39, 56]). The problem is well-known to be NP-complete (see for instance [50, 53]), and often, only a good solution can be obtained, while it is difficult to ensure optimality. We will focus

here on instances where the optimal tour is known.

Various instance collections exist that can be used to benchmark solution algorithms. In this thesis, we will consider the library TSPLIB [55] that collects instances from various sources, some randomly generated, some collected from specific applications (see for instance Johnson and McGeoch [36], and section 5.1). The optimal solution is known for various instances, and is given in the library, allowing comparisons.

4.2 Tour Construction Heuristics

Heuristics can be used to generate promising tours that can be later used as starting solutions for optimisation algorithms. The two most important factors in the tour construction are the time needed to create the tour and its solution quality in terms of tour length. Several heuristics have been proposed, with specific features, as described for instance in [36] and ([56] chapter 6, p. 73). In this thesis, we will use the nearest neighbor algorithm to produce the initial solution.

The simplest way to build a tour is consider the greedy algorithm 2. The tour is constructed in an incremental way, adding to the last city in the tour the nearest neighbor in the set of unvisited cities. The initial city can be set to 1, as in algorithm 2, or selected at random. When all the cities have been visited, we close the tour by returning to the initial city. The algorithm is in $\Theta(\mathbb{N}^2)$ [56, 61]. The figure 4.1 illustrates the method.

Algorithm 2 Nearest neighbor algorithm

1: $tour \leftarrow (1)$. Set $T = \{2, \ldots, \mathbb{N}\}$ and $l = 1$.
2: While $T \neq \emptyset$ do the following.
3: Select $j \in T$ such that $d(l, j) = min\{d(l, i) \mid i \in T\}$.
4: Connect l to j, $tour \leftarrow (tour, j)$. Set $T \leftarrow T \setminus \{j\}$ and $l = j$.
5: Connect l to the 1 to form a tour and set $tour \leftarrow (tour, l)$.
6: **return** *tour*

4.3 Moves

In order to apply SA algorithm to solve the TSP, we consider a tour as a solution and the associated energy as the tour length. The tour consists of the sequence of cities

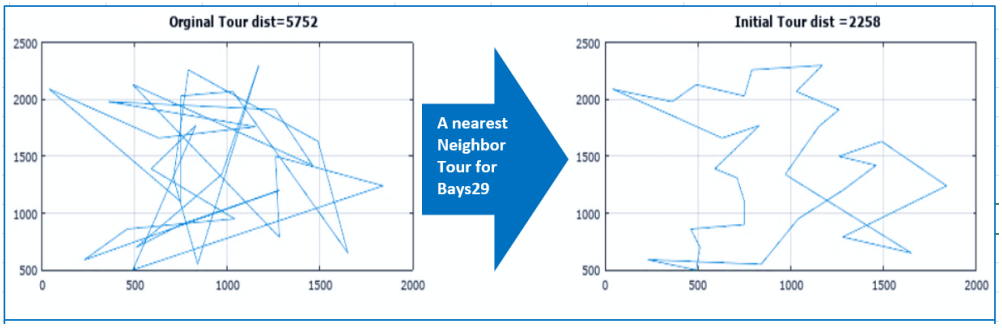


Figure 4.1: Tour produced by the nearest neighbor algorithm for problem bays29 from TSPLIB

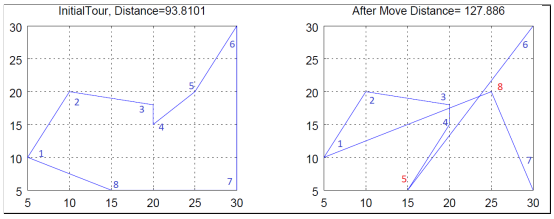
indexes, each city being represented by a set of two coordinates. The SA algorithm requires the generation of a candidate solution in the neighborhood of the current solution at each iteration. This can be achieved by applying a move or a set of moves to the current solution that reorder the sequence of visits [56, 61]. We only consider here moves applied within a tour, illustrated in figure 4.2, while there exists other moves defined between several tours (see for instance [61]). By convention, we will denote the predecessor of city i in a tour by i^-, and its successor by i^+.

The simplest moves consist to permute two cities in the sequence of visits. We can choose such cities as follows.

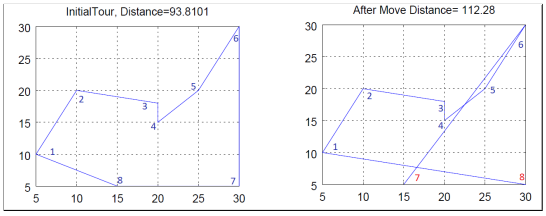
Select-Pos-1-Random-OPt This move consist to randomly select two different locations, in the same tour, and swap them. For instance, given the initial tour is 123456781, we could select the cities 5 and 8. Swapping them produces the new tour 123486751, as illustrated in figure 4.2a.

Select-Pos-1-Previous-OPt We can simplify the move by selecting only one city at random and swapping it with its predecessor in the tour. For instance, if we select the city 8 in the tour 123456781, we will swap it with the city 7, leading to the tour 123456871, as in figure 4.2b.

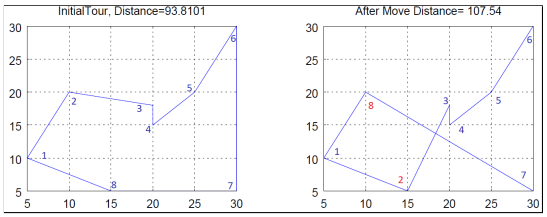
Select-Pos-1-Next-OPt Similarly, we can swap a city with its successor in the tour. If we select the city 8, we will swap it with the city 2, as in figure 4.2c.



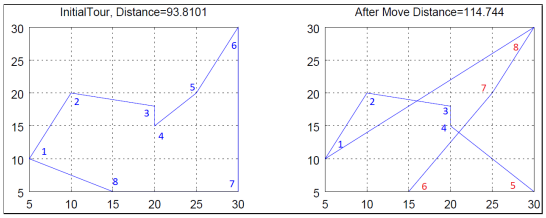
(a) Select-Pos-1-Random-OPt



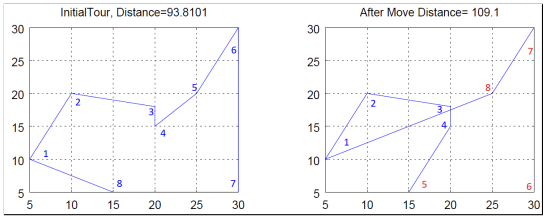
(b) Select-Pos-1-Previous-OPt



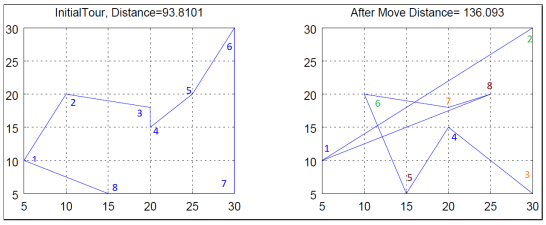
(c) Select-Pos-1-Next-OPt



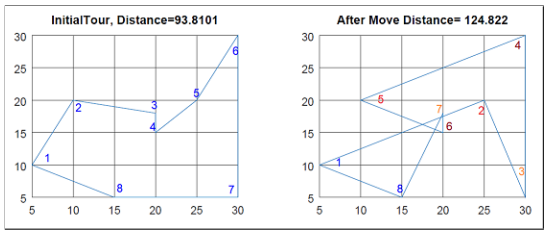
(d) Select-Pos-2-1-OPt



(e) Select-Pos-2-2-OPt



(f) Select-Pos-3-1-OPt



(g) Select-Pos-3-2-OPt

Figure 4.2: Several types of moves

Instead of permuting cities, we can switch edges, keeping their original directions and reversing them, as explained below. We can characterize this kind of move by the general denomination Lin-2-Opt.

Select-Pos-2-1-OPt Pick i, j at random such that $i \neq j$, $j \neq i^-$, $i \neq j^+$. Then, do $swap(i^-, j)$ and $swap(i, j^+)$. For instance, as in figure 4.2d, if we select cities 6 and 7 in the tour 123456781, the move is equivalent to switch the edge (5,6) and the edge (7,8), leading to the tour 123478561.

Select-Pos-2-2-OPt Pick i, j at random such that $i \neq j$, $i^- \neq j^+$, and do $swap(i, j)$, $swap(i^-, j^+)$. For instance, as in figure 4.2e, if we select cities 6 and 7 in the tour 123456781, the move is equivalent to switch the edge (5,6) and the edge (7,8) and reverse them, leading to the tour 123487651.

The previous moves can be generalized to the exchange of more nodes, by composing edges exchanges or reversing edges. This leads to Lin-n-Opt moves, where n is the number of edges involved. We will consider two Lin-3-Opt moves. More general moves can be found in [61].

Select-Pos-3-1-OPt Pick i, j, k, such that $i \neq k$, $i^- \neq j^+$, $j \neq k^+$. Do swap (i^-, j^+), $swap(i,k)$, $swap(j,k^+)$. For instance, if we select $i = 3$, $j = 5$, $k = 7$, the initial tour 123456781, from the graph 4.2f, we can see the initial by reversing 2-6, 3-7, 5-8 to become 167482351.

Select-Pos-3-2-OPt Pick i, i^-, j, j^+, k, k^+ such that $1 \leq i, j, k \leq \mathbb{N}, i^- = i - 1, j^+ = j + 1, k^+ = k + 1, swap(i^-, j^+), swap(i, k^+), swap(j, k)$. For instance, if we select $i = 3$, $j = 4$, $k = 6$, from the graph 4.2g, we can see the initial tour 123456781 by reversing 2-5 , 3-7, 4-6 to become 157624381.

Finally, we can mix the moves together and select the combination that deliver the tour with the least cost. According to Černỳ [17], it is not possible to determine the best type of moves for a given instance, and we have to proceed by trial and error. As a heuristic, we can decide to accept a move only if it produces a tour of smaller

length, but as noted by Bertsimas and Tsitsiklis [12], it is often more efficient to generate several consecutive moves, even if some individual moves lead to a longer tour, as the combination can result in a better tour, as done in the heuristic proposed by Lin and Kernighan [44]. In a context like the SA algorithm, in the diversification phase, we will admit any move to produce a new tour, that will be accepted or rejected on the basis of the metaheuristic logic.

4.3.1 Ruin and Recreate

According to Schneider and Kirkpatrick [61], considering small moves only may not always be adequate as in some cases, it is not easy to escape from a poor local minimizer; therefore, making larger moves in the tour, producing a big change in the tour construction, may be beneficial. A popular strategy is the ruin and rebuild technique, consisting first in the destruction of the tour or part of it (ruin), removing some parts of the tour randomly, and next in the construction of a new tour (rebuild) reinserting the removed parts using some construction heuristics and keeping the remaining of the tour untouched. We will however not investigate further this approach in this thesis.

4.4 Permutations

When the number of cities is not too large, it is possible to compute all the solutions and return the optimal one. For instance, if there are 6 cities to visit, the are 5! feasible solutions, that can be obtained by generating all permutations of cities 2 to 6 (recall that city 1 is fixed as the origin and end of the tour). This allows us to easily compare the obtained solution by some optimization algorithm to the optimal tour.

CHAPTER 5

INITIAL EXPERIMENTATIONS

5.1 TSP Instances

In order to validate our SA algorithm implementation, we test it on various TSP instances. We first create a toy problem with 8 cities randomly generated on a two-dimensional space, and use the Euclidean distance to compute the tour length. We call this problem *rnd8*. The others problems are taken from TSPLIB [55] and presented in table 5.I. The distance between the cities can be explicitly stored in a matrix, possibly in triangular form in case of symmetric distances, or given implicitly, by simply storing the cities coordinates, the distances being computed using the Euclidean distance. More information can be found in Reinelt [55].

No	Name	# Cities	Metric	Distance format
1	eil51	51	2D Euclidian	Not Explicit
2	pr76	76	2D Euclidian	Not Explicit
3	eil101	101	2D Euclidian	Not Explicit
4	pr107	107	2D Euclidian	Not Explicit
5	bier127	127	2D Euclidian	Not Explicit
6	a280	280	2D Euclidian	Not Explicit
7	bays29	29	Geograhical	Full matrix
8	gr17	17	Explicit	Lower diagonal matrix
9	gr24	24	Explicit	Lower diagonal matrix
10	gr21	21	Explicit	Lower diagonal matrix

Table 5.I: TSP instances

We pre-process the instances by computing distances matrices, so that we do not have to recompute the distances during the optimization process, and we identify the cities by their indexes $1,\ldots,N$. The SA algorithm can be repeated n times, using as starting solution the last solution found at the end of previous SA execution. We can store the best and final solution for each execution as well as the best overall solution. Finally, we use the random number generator "MRG32k3a" [42] in our project.

5.1.1 Rnd8 Problem

Our toy problem consists of 8 cities, with coordinates $(5,10)$, $(10,20)$, $(15,5)$, $(20,15)$, $(25,20)$, $(30,30)$, $(20,18)$, $(30,5)$. We first compare the moves described in chapter 4 to determine neighbor solutions, using a linear decreasing temperature, with $T_i = 50$, $T_f = 0.00001$ and a cooling rate α of 0.1. We perform 21 SA replications using MH acceptance technique. The results are reported in table 5.II while we represent the evolution of final solution over the replications in figure 5.1. In this experiment, the move providing the best results is (Select-Pos-3-2-OPt) while the worst is (Select-Pos1-Previous-Opt).

No	Type of move	**Mean final cost**	Final tour	Mean best cost	Best tour
1	Select-Pos-1-Random-OPt	**93.21**	187435621	93.10	187653421
2	Select-Pos-1-Previous-OPt	**97.83**	126534781	93.15	187653421
3	Select-Pos-1-Next-OPt	**94.43**	123654781	93.10	187653421
4	Select-Pos-2-1-OPt	**96.37**	124365781	93.12	187653421
5	Select-Pos-2-2-OPt	**95.19**	124365781	93.12	187653421
6	Select-Pos-3-1-OPt	**93.30**	187435621	93.15	187653421
7	Select-Pos-3-2-OPt	**93.18**	**124365781**	**93.10**	**187653421**

Table 5.II: Comparison of moves on problem rnd8

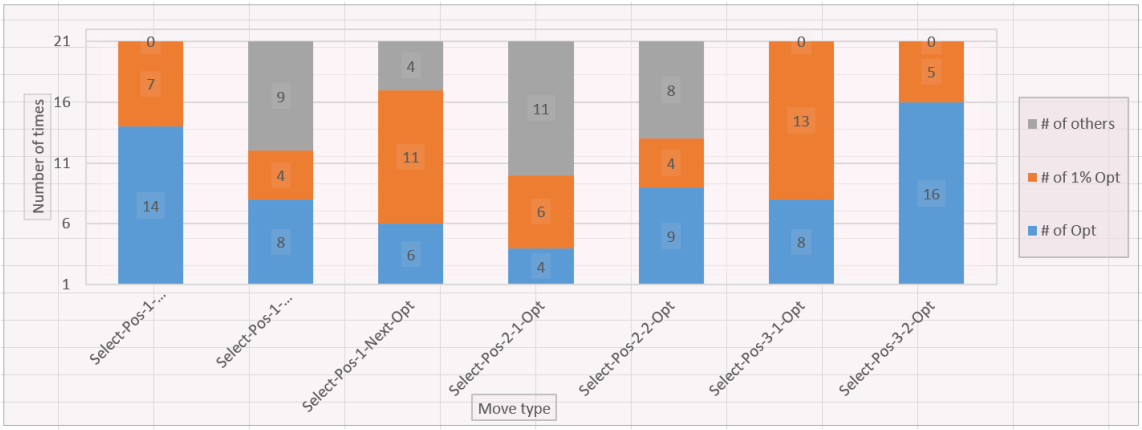


Figure 5.1: Results based on final distances for different move types

We next run the SA algorithm with parameters described in table 5.III, using a linear temperature scheme, $n = 21$, and $r_k = 8$ for all k.

Acceptance Type	Move Type	Variance	T_i	T_f	Cooling Rate (α)
Metropolis	Select-Pos-2-2-opt	0	100	0.001	0.001

Table 5.III: Parameters for toy problem rnd8

Figure 5.2 reports the best and final solutions found over the SA executions. The average best tour distance over the 21 executions equals to 93.10 and the average final tour distance over the 21 executions is 93.49. Figure 5.3 represents the original tour when we visit the cities in the order of their indexes, the initial tour obtained using the greedy heuristic, the best overall tour and the optimal tour determined by computing all possible permutations. The associated costs are given in 5.IV. Finally, figure 5.4 illustrates the minimum required (Select-Pos-2-2-opt) moves to attain the optimal solution from a given final solution at the previous SA execution.

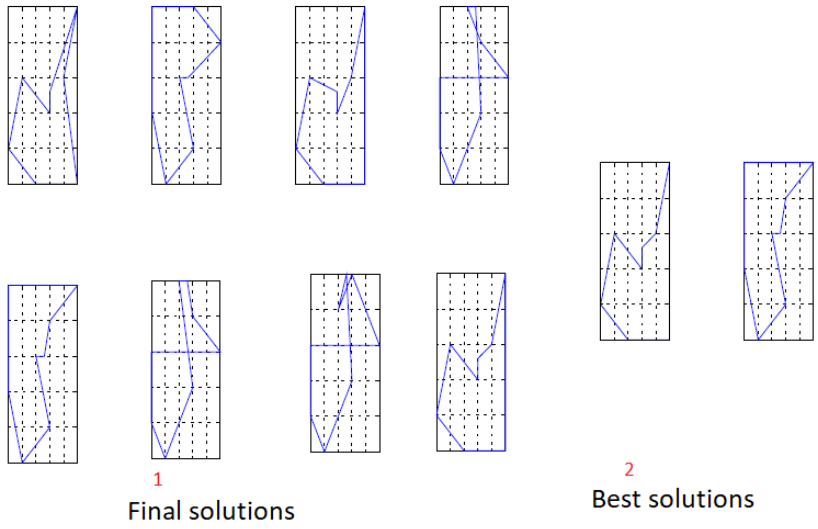


Figure 5.2: Final and best solutions for rnd8 problem

	Original	Initial	Overall best	Final	Optimal
Cost	113.94	93.81	93.10	93.10	93.10
Order	123456781	127456831	187653421	124356781	187653421

Table 5.IV: rnd8 tour costs

The experiment shows that as the initial temperature is high, many bad solutions are accepted, moving away from optimality, and if the temperature is not sufficiently

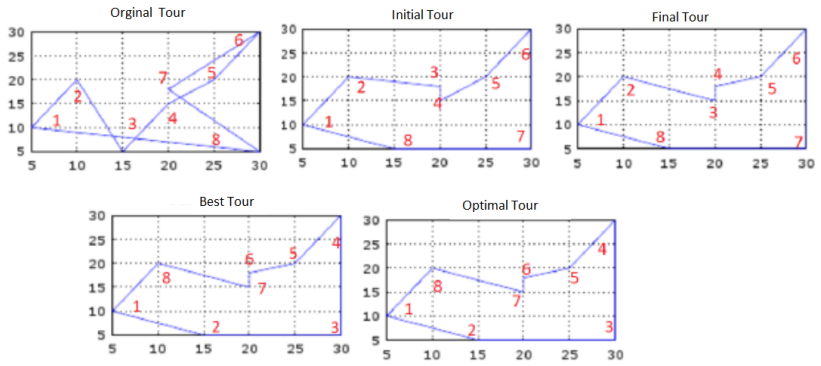


Figure 5.3: rnd8 tours

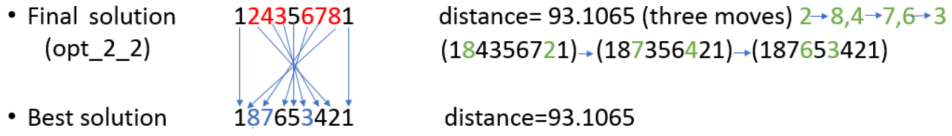


Figure 5.4: Minimum required moves needed to pass from final to best solution

reduced during the SA execution, the final tour can have a too large value and its corresponding graph presents many crosses, as shown in figure 5.2. In addition, the best solutions are dominated by the optimal solution. Restarting the algorithm *n* times, using the final tour at a given execution as the starting tour for the next one, allows to get better performance. The final and best solutions tend to improve over the executions, as sometimes, the algorithm gets stuck at a bad local minimum at low temperatures, but restarting SA algorithm allows to escape from it. However, restarting the algorithm presents some drawbacks too, as at the beginning of each execution, many uphill moves can be accepted, degrading the solution quality, even when only a few moves would have been sufficient to reach the optimal solution, as illustrated in figure 5.4.

5.2 Temperature management

Previous experiments suggest that the choice of initial and final temperatures, as well as the cooling rate, significantly impacts the performance of SA. We explore the sensitivity to the temperature cooling approach in more details in this section.

5.2.1 Temperature Schemes Comparisons

Table 5.V summarizes experiments performed of rnd8 for different choices of temperatures and cooling rates (α) as defined in page 12, using the move (Select-Pos-3-2-opt) as empirically, it delivers the best performance. In addition, the linear temperature update scheme is used and $r_k = 8$. The classification type suggests various possible choices for the parameters, but highlights the challenge to fix them despite the simplicity of the example rnd8. Consequently, there is a need to find a mechanism that automates the choice of parameters in a sensible way.

No	Classification Type	T_i	T_f	α	**avg Final Cost**	avg Best Cost
1	high T_i	100	0.00001	0.1	**93.26**	93.10
2	low T_i	10	0.00001	0.1	**93.16**	93.10
3	medium T_i	50	0.00001	0.1	**93.18**	93.10
4	low T_f	50	0.01	0.1	**93.22**	93.10
5	very low T_f	50	0.000001	0.1	**93.22**	93.10
6	T_f close to Zero	50	0.00000001	0.1	**93.24**	93.10
7	high α	50	0.00001	0.8	**96.76**	93.10
8	low α	50	0.00001	**0.00001**	**93.10**	**93.10**
9	medium α	50	0.00001	0.01	**93.19**	93.10

Table 5.V: Comparison of temperature update schemes for rnd8, using move Select-Pos-3-2-opt and $n = 21$

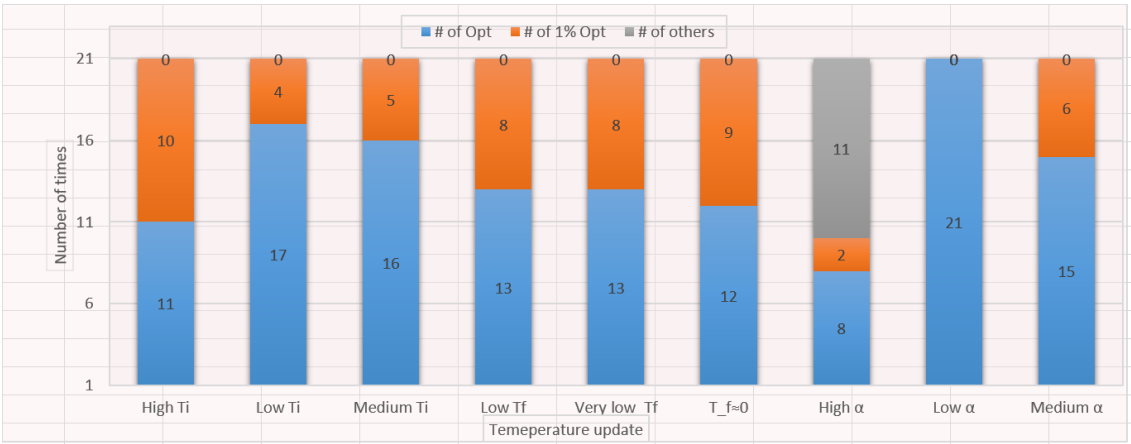


Figure 5.5: Final tour distances for $n = 21$ SA executions, based on temperature choices

From the table 5.V, we can see that the best solution for the toy problem rnd8 is obtained with experiment 8 and the worst with experiment 7. They correspond to the

slowest and the fastest cooling rates, respectively, exhibiting that SA algorithm needs time to stabilize on the correct solution. As observed in figure 5.5, in case of high temperature, any move is accepted, while in low temperature only downhill moves are accepted, and SA acts as a local search. Similarly, when the cooling rate is fast, the diversification phase takes place for a limited time only and the SA reduces to a local search.

5.2.2 Initial and Final Temperatures Selection

This section represents how to select the temperature parameters that will be used in the simulation. As illustrated in table 5.V and mentioned by van Laarhoven and Aarts [41], the choice of initial and final temperatures strongly impact(s) the algorithm efficiency. We slightly adapt in algorithm 3 the method proposed by Ben-Ameur [11], van Laarhoven and Aarts [41], assuming that the acceptance probability follow(s) the Metropolis-Hastings criterion, as described in section 2.2.1. For any state x_i and some neighbor x_j, we record the energy difference from the lowest energy state to the highest energy state, and repeat the procedure for ν iterations. We then compute the average energy difference over the ν observations, and deduce from it approximate values of initial temperature T_i and final temperature T_f, corresponding to target probability levels P_1 and P_2, respectively, of uphill move acceptance. Table 5.VI illustrates the temperatures selection on several TSPLIB instances, described in table 5.I, page 30 . We can observe that the initial and final temperatures greatly vary over the problems, illustrating the need to properly choose them on an instance basis. The configuration of cities has a huge impact on the initial and final temperatures, but more importantly, the average cost provides a better indication of the parameter values to use, that are important to ensure the progress of the algorithm towards a good solution. At high temperature, we want to favor the exploration of the solutions space, so P_1 should be close to 1, following the recommendation made by Saıt and Youssef [60] that the initial temperature should be

selected such that:

$$\frac{\text{Number of moves accepted at } T_0}{\text{Total number of moves attempted at } T_0} \simeq 1$$

At a low temperature, we tend to reject any uphill move, suggesting a value of P_2 close to 0. However, too high P_1 could cause the algorithm to act as a random search algorithm for many iterations, while a too small P_2 value will often lead the algorithm to stagnate as the probability to accept an uphill move is then very small, while typically the solution cannot be improved locally.

Algorithm 3 Selection of initial and final temperatures

1: Set $m = 0$, $\nu > 0$, and probability levels $P1$ and $P2$.
2: **while** $m < \nu$ **do**
3: Generate a random solution x_i and a neighbor solution x_j.
4: $\Delta E_m \leftarrow |E(x_i) - E(x_j)|$
5: $m \leftarrow m + 1$
6: **end while**
7: $\overline{\Delta E} = \sum_{m=1}^{\nu} \Delta E_m$
8: $T_i = -\overline{\Delta E} / \log P_1$
9: $T_f = -\overline{\Delta E} / \log P_2$

			$P_1 = 0.9$	$P_2 = 0.00001$	$P_1 = 0.8$	$P_2 = 0.001$	
No	Size	Name	T_i	T_f	T_i	T_f	$\overline{\Delta E}$
1	8	rnd8	**104.31**	0.96	**49.25**	1.59	**10.99**
2	51	eil51	**389.32**	3.6	**183.83**	5.94	**41.02**
3	76	pr76	**93626.8**	856.83	**44207.3**	1428.04	**9864.57**
4	101	eil101	**430.90**	3.94	**203**	6.57	**45.40**
5	107	pr107	**95010**	869.49	**44860.4**	1449.14	**10010.3**
6	127	bier127	**53762.2**	492.00	**25384.6**	820.00	**5664.41**
7	280	a280	**1784.03**	16.33	**842.4**	27.21	**187.96**
8	29	bays29	**2430**	22.23	**1147.43**	37.06	**256.04**
9	17	gr17	**2971.47**	27.19	**1403.02**	45.32	**313.07**
10	24	gr24	**1739.18**	15.91	**821.17**	26.52	**183.24**
11	21	gr21	**4063.97**	37.19	**1918.86**	61.98	**428.18**

Table 5.VI: Selection of initial and final temperatures on different TSPLIB instances ($\nu = 100000$)

5.3 Comparison of acceptance functions

We first compare MH and Glauber acceptance functions on our toy problem rnd8. The initial temperature T_i is set to 50 and the final temperature T_f, to 0.001. We use the move Select-Pos-3-2-OPt, a linear temperature decrease with a cooling rate α of 0.001 as defined in section 2.2.3, and perform 21 SA executions. In terms of average final and best solutions, both approaches perform similarly, delivering a tour with the optimal length of 93.10. Figure 5.6 exhibits the cost evolution over the iterations of one SA execution, for the temperature range 2–0.001. The initial temperature T_i is set to 50, but we do not plot the iterations in the temperature range 50–2 due to the high volatility of the process for large temperatures. We can observe that the Glauber acceptance function leads to more volatility of the solution than the MH acceptance, but both methods stabilize and converge to the optimal cost when the iteration index increases. Figure 5.7 illustrates some of the generated tours.

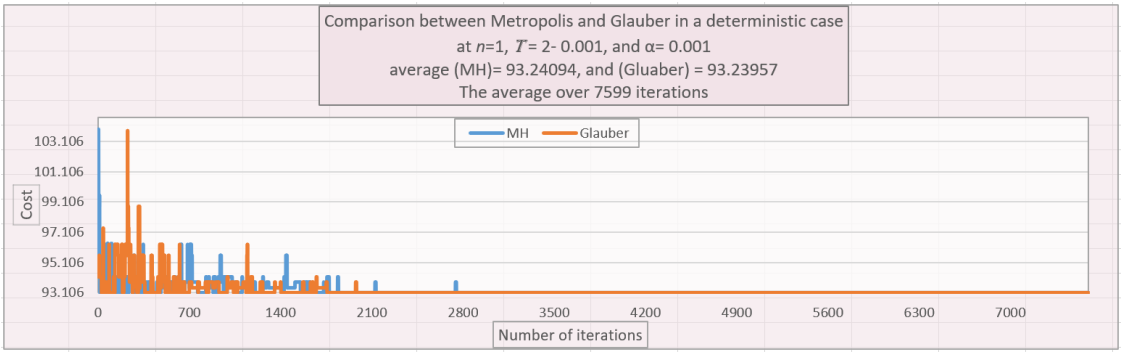


Figure 5.6: Comparison of Glauber and MH at low temperature for rnd8

The experiments have been reproduced on various TSPLIB instances, with an initial temperature T_i set to 50, 100 or 1000, based on pilot tests, and we use 3 SA replications. The final temperature T_f is set to 0.0001 and the cooling rate to 0.001. The results are reported in table 5.VII. They exhibit that while enjoying a strong convergence theory, SA often encounters practical difficulties to find the optimal solution with MH and Glauber acceptance mechanisms. We have however identified four problems for which the algorithm finds the optimal tour.

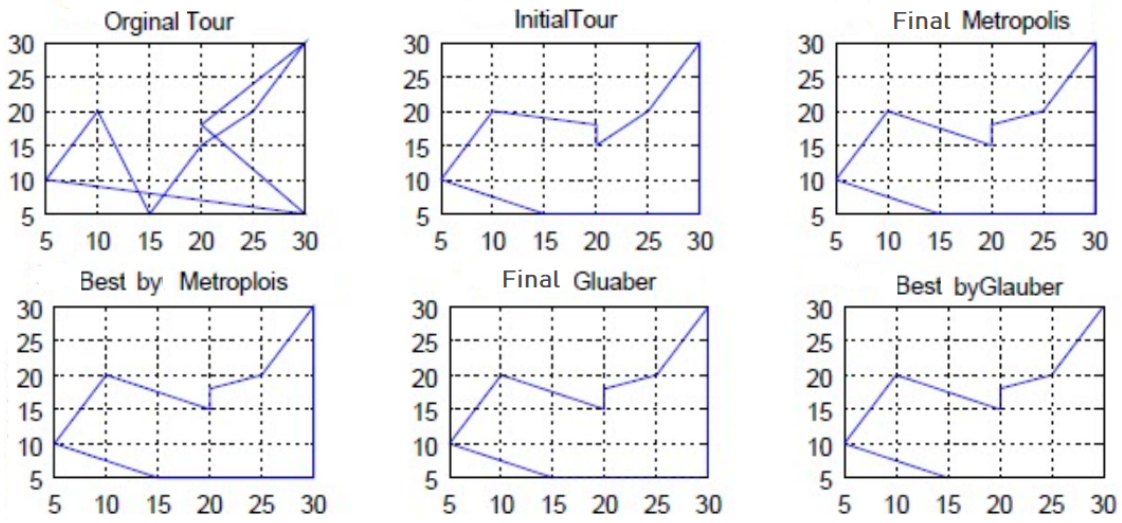


Figure 5.7: Tours generated by SA with MH and Glauber acceptance functions

No	Name	Initial Solution	**Best (MH)**	**Best (Glauber)**	Optimum
1	eil51	513.61	**444.81**	**441.45**	426
2	St70	805.53	**779.78**	**825.24**	675
3	eil101	938.31	**938.31**	**938.31**	629
4	pr107	46678.2	**44857.9**	**44857.9**	44303
5	rat99	1564.72	**1555.61**	**1542.06**	1211
6	bier127	135752	**125046**	**126081**	118282
7	a280	3148.11	**3148.11**	**3148.11**	2586
8	berlin52	8980.92	**8210.56**	**8323.07**	7542
9	pr76	153462	**114837**	**110332**	108159
10	**gr21**	**3333**	**2707**	**2707**	**2707**
11	**gr17**	**2187**	**2085**	**2085**	**2085**
12	**gr24**	**1553**	**1272**	**1272**	**1272**
13	**bays29**	**2258**	**2020**	**2020**	**2020**

Table 5.VII: Solutions found by SA using Glauber and MH acceptance for TSPLIB instances

5.4 Experimental Results of SA Under Noise

As in (3.4), we assume that the cost (energy) of a solution can be decomposed as the sum of the mean cost and a white noise:

$$E(x) = \overline{E}(x) + \varepsilon(x, \omega),$$

where $\varepsilon(x, \omega)$ is i.i.d. over the feasible solutions and $\varepsilon(x, \omega) \sim N(0, \sigma^2)$. In our experimentations, we simulate the noise using Monte Carlo draws, and add it to the (deterministic) tour cost.

We consider the problems rnd8, gr17, gr21, and gr24, for which SA can find the optimal solution in the deterministic case, with variance 0.3 and 2. Both MH and Glauber acceptance procedures are tested, with a different number of simulations among the problem, and a linear temperature decrease. The experimental configurations are summarized in table 5.VIII. It can be noticed from the figure 5.10 that in the example case gr17, Glauber and MH acceptance mechanisms behave in a similar way regarding the final and best solutions, whereas in the other examples the methods behave differently. More importantly, the figure reveals that the returned best cost underestimates the expected best cost and the final cost is not always stable. Without noise correction, SA provides biased solutions and has more difficulty to stabilize. It is therefore important to modify SA to take the noise into account, as in the following chapters.

Name	n	Acceptance Type	Move Type	Variance	T_i	T_f	Cooling Rate
rnd8	21	MH/Glauber	opt3-2-opt	0.3	50.0	0.00001	0.00001
gr17	15	MH/Glauber	opt3-2-opt	0.3, 2	50.0	0.001	0.01
gr21	101	MH/Glauber	opt3-2-opt	0.3	90.0	0.0001	0.01
gr24	101	MH/Glauber	opt3-2-opt	0.3, 2	100.0	0.00001	0.001

Table 5.VIII: Parameters for noisy problem examples

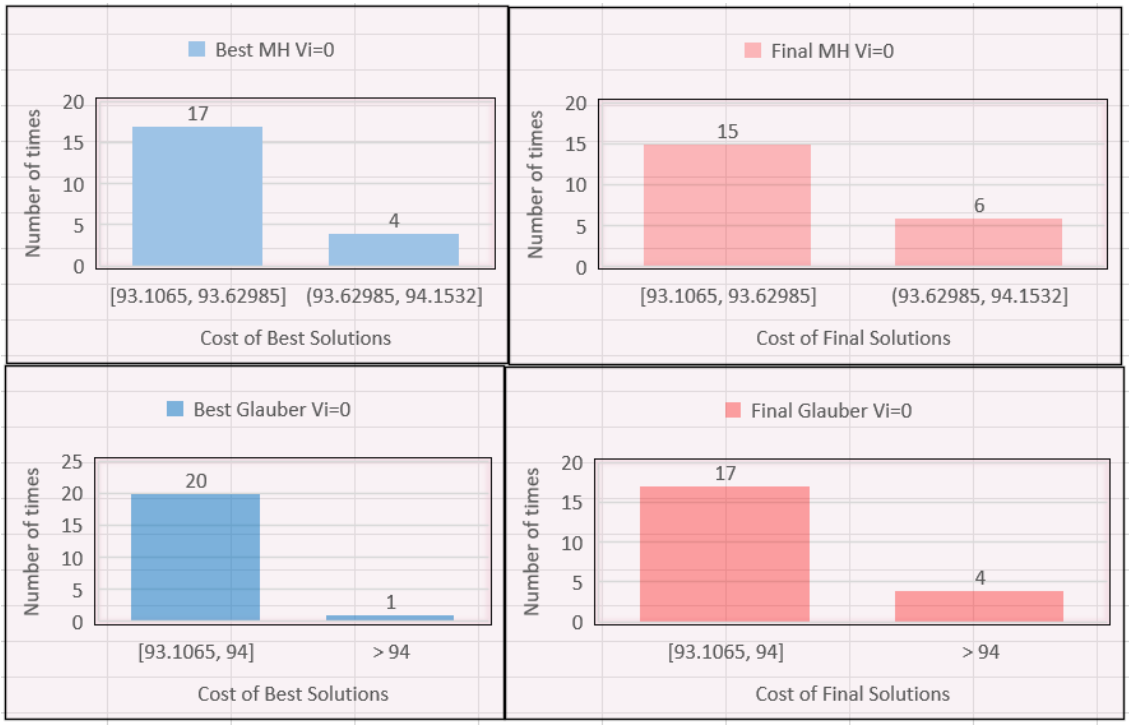


Figure 5.8: Noisy Rnd8

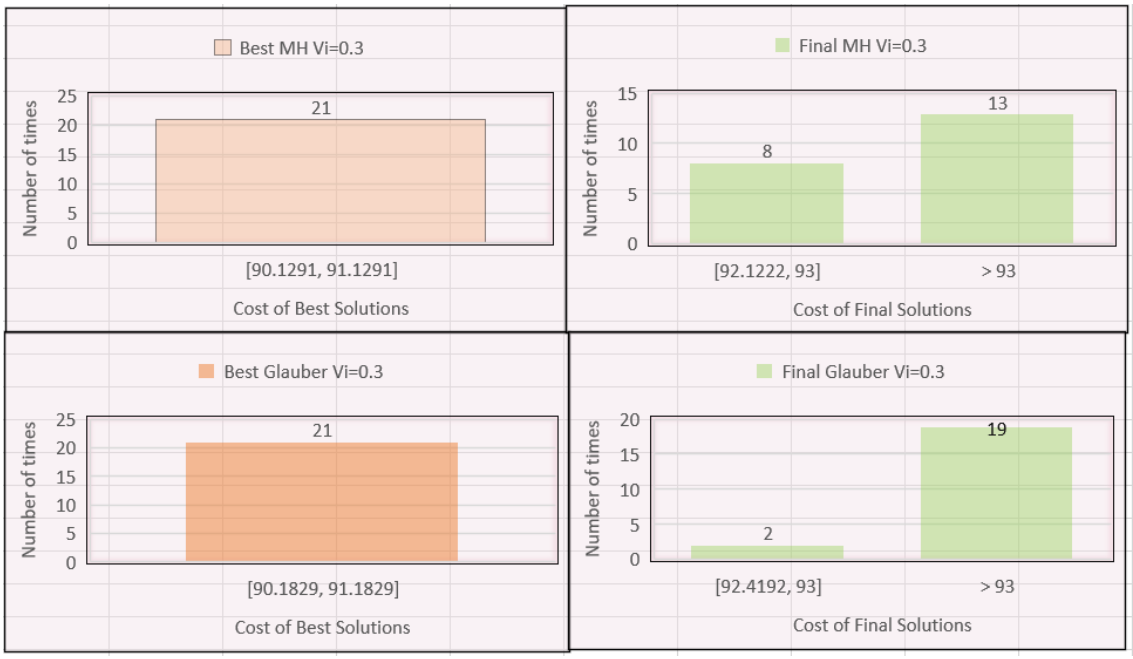


Figure 5.9: Noisy Rnd8

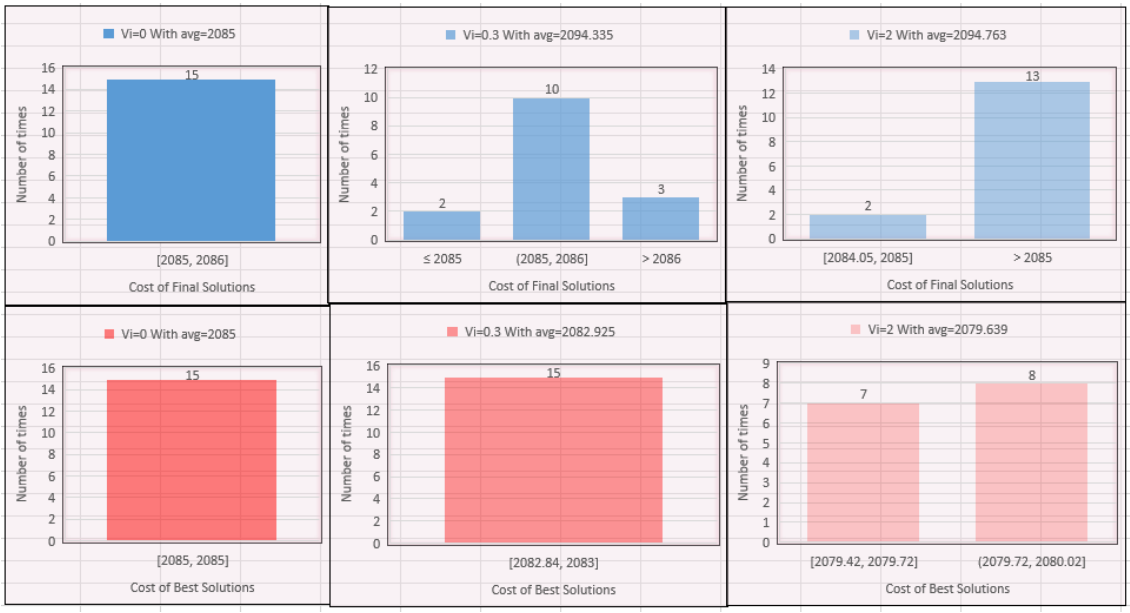


Figure 5.10: Noisy gr17 for MH and Glauber

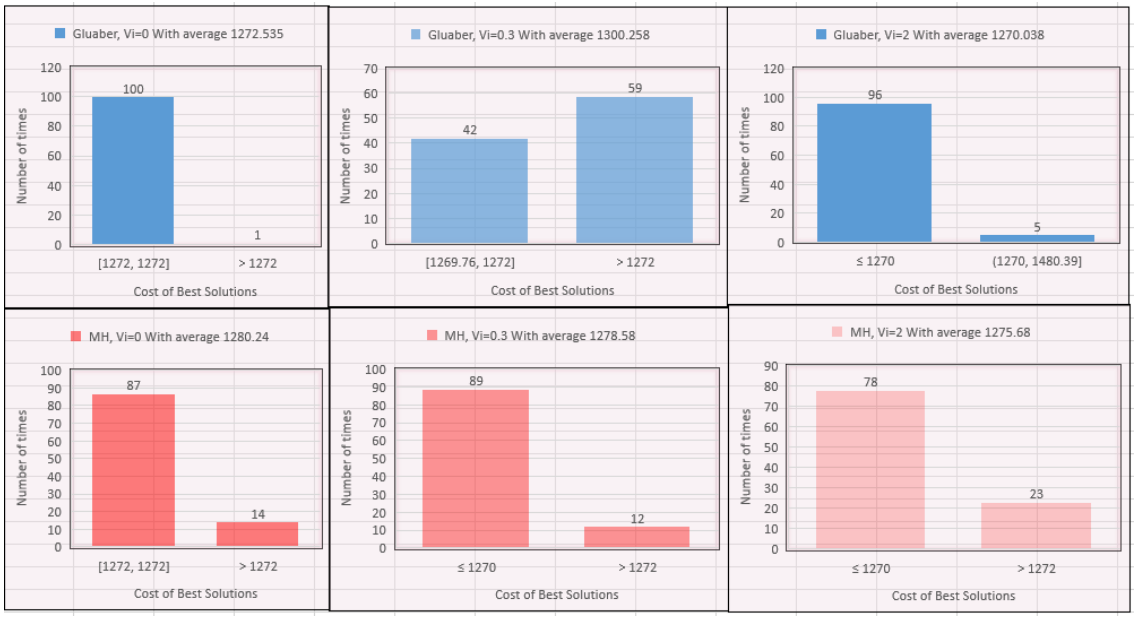


Figure 5.11: Best noisy costs for gr24 MH and Glauber

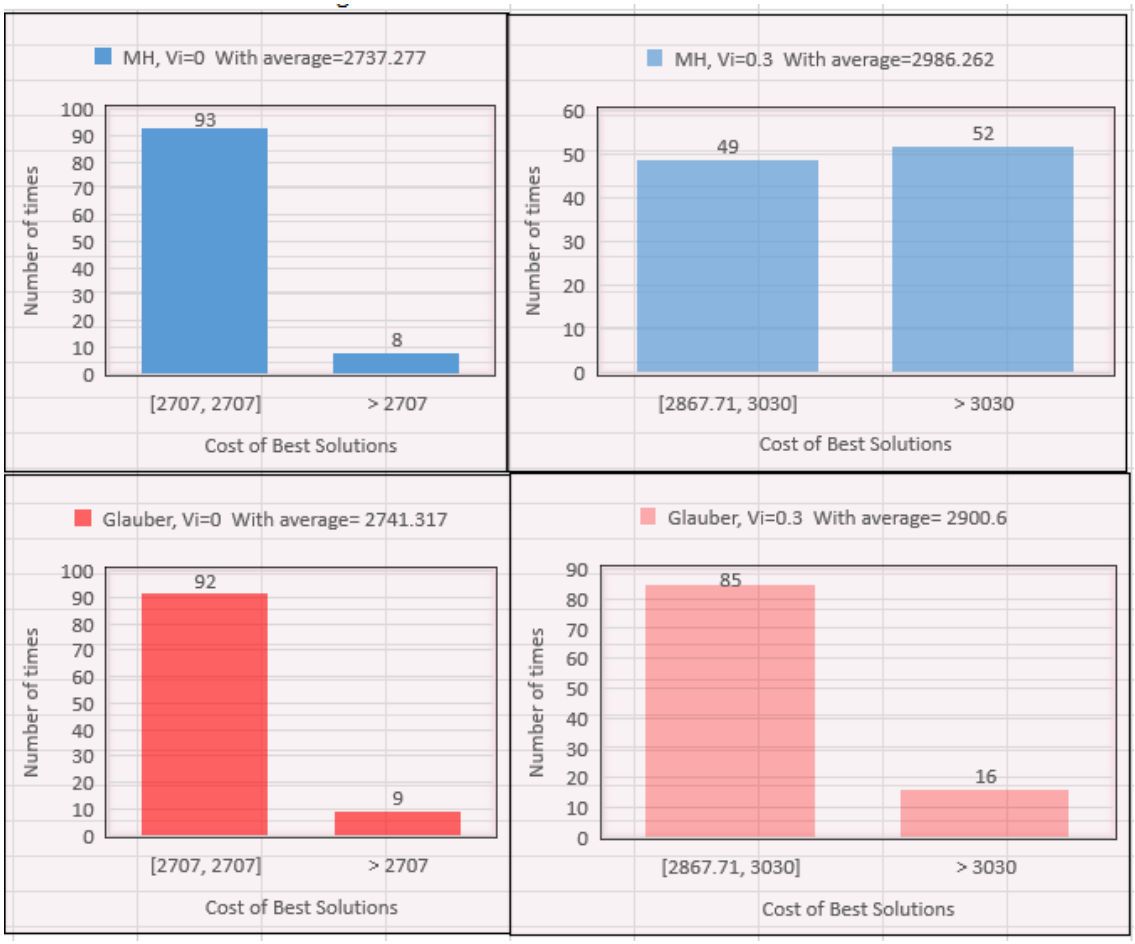


Figure 5.12: Best noisy costs for gr21 MH and Glauber

CHAPTER 6

COMPARISONS OF NOISE MANAGEMENT STRATEGIES

In the previous chapter, we have analysed SA algorithm behavior when solving deterministic and stochastic TSPs. We will now compare various simulated annealing under noise (SAUN) strategies in terms of computation cost and quality of the solutions. Due to the stochastic nature of the methods and the problems, we will study their average behavior on several instances, as follows. For each instance, we run each method 200 times and record the final solutions. We record the fraction of runs where the optimal solution was reached and where a near-optimal solution was obtained, as well as the average final tour length in deterministic and noisy cases. Near-optimality is defined as a tour length no more than $(1+\varepsilon)$ times the optimal cost. In our experiments, we set ε to 1%.

In order to keep the computational times reasonable, the noise was artificially generated as a Gaussian noise scaled to the required variance level, and added to the tour length without noise. The computation cost was then scaled to reflect the number of repetitions that would have been needed to obtain the same variance reduction.

The results are graphically compared to facilitate the discussion, using a procedure inspired by the method proposed by Dolan and Moré [21] to benchmark optimization software. For each approach, we create a curve capturing in y-axis the proportion of simulations that reach optimality and a curve for near-optimality, with respect to the computational effort, represented in logarithmic scale in x-axis, where one computational cost unit corresponds to one tour evaluation. In other terms, we monitor the evolution of the algorithms as the computational effort increases, and analyse their efficiency in terms of effort required to reach good solutions as well as their robustness, measuring their capacity to asymptotically reach (near-)optimality.

6.1 Experimental Settings

The choice of initial and final temperatures for SAUN is considered in this section. We here rely on the procedure proposed in section 5.2.2 for the deterministic case to make a first proposition of these temperatures, but refine them as this approach gives rough estimates of the temperatures, so the real acceptance probabilities differ from the targets, as reflected in table 6.I, where we report the proportion of accepted uphill moves over 200 experiments, using MH acceptance criterion, on various problem instances and various initial temperatures. In this table, N_a^{det} and P_a^{det} corresponds to the number and the proportion of accepted moves, respectively, in the deterministic case, while N_a^{stoch} and P_a^{stoch}, while the latest column indicates the corresponding target probability in algorithm 3. We observe that the acceptance probabilities are then always underestimated. On the basis of this finding, we have decided to divide the initial temperature by two as long as the empirical probability is higher than the desired probability in order to limit the exploration phase and facilitate the algorithm capability to attain a low final temperature in a reasonable time. The same approach can be applied to deterministic and stochastic cases, and for any acceptance scheme. While heuristic, this technique has proved effective in our numerical experiments.

No	T_i	N_a^{det}	N_a^{stoch}	P_a^{det}	P_a^{stoch}	$e^{\frac{[\Delta E]}{T_i}}$
1-rnd8	10.5	95	81	0.47	**0.40**	0.35
2	15	121	110	0.60	**0.55**	0.48
3	20	140	135	0.7	**0.65**	0.57
4	25	161	151	0.80	**0.75**	0.64
5	30	165	159	0.82	**0.79**	0.69
6	35	169	166	0.84	**0.83**	0.73
7	49.25	185	172	0.92	**0.86**	0.80
8-gr17	1403	178	178	0.89	**0.89**	0.80
9	701.5	150	152	0.75	**0.76**	0.63
10	350.75	98	98	0.49	**0.49**	0.40
11-bays29	1147	184	182	0.92	**0.91**	0.79
12	573.5	153	157	0.76	**0.78**	0.63
13	286.75	104	103	0.52	**0.51**	0.40

Table 6.I: Acceptance of uphill moves for problems rnd8, gr17, and bays29

We use logarithmic temperature update scheme, setting the temperature at iteration k as

$$T_k = \frac{c}{\log(k+d)},$$

where $c = T_i \log 2$ and $d = 1$, or

$$T_k = \frac{c}{\log(kd)},$$

where d is a constant greater than 1.

Even if the methods are designed to lower random noise as the temperature decreases, they still can face difficulties to converge when the original noise is too large, as the noise tends to dominate in the objective function evaluation at each iteration. A possible way to control the noise is to require that a 95% confidence interval over the optimal tour length has a half-width no more than 10% than this length. For instance, the optimal length of our toy problem *rnd8* is approximately 93, and rounding the 0.975 quantile of a $N(0,1)$ to 2, this implies that the initial standard deviation should be no more than 4.5. The computational cost is simulated by normalizing it at 1 with the original variance, and at iteration k, its value corresponds to the variance reduction factor with respect to the original variance. The variance at the iteration k is noted σ_k^2. CD acceptance rate is sensitive to the noise level as a higher variance will result in a lower acceptance probability, that can be arbitrarily small when the variance grows, as observed by Branke et al. [15]. It is therefore important to control the initial variance level, setting

$$\sigma_0^2 = \frac{\sigma_i^2}{\nu},$$

where $\nu > 0$. Enforcing (3.7) as an equality, that is

$$\sigma_{\Delta E}^2 = \frac{2\sigma_0^2 T^\eta}{T_0^\eta},$$

we can set the initial variance σ_0^2 in various ways. For instance, if we fix the value σ_f^2, we can set ν such that

$$2\sigma_f^2 = \frac{2\sigma_i^2 T_f^\eta}{\nu \sigma_f^2 T_0^\eta},$$

or

$$\nu = \frac{\sigma_i^2 T_f^\eta}{\sigma_f^2 T_0^\eta}.$$

Table 6.II illustrates some values obtained using this approach. A more advanced strategy is to choose ν in order to have the acceptance probability, computed as in (3.6), close to some predefined threshold α, that is we search a value ν such that

$$P_a = \mathbb{E}[P_{ij}] = \mathbb{E}\left[P\left[\Delta E_{ij} \leq -\frac{\sigma_i^2}{\nu T}\right] + P[\text{Accept}]P\left[\Delta E_{ij} \geq -\frac{\sigma_i^2}{\nu T}\right]\right] = \alpha,$$

where the expectation can be estimated using a Monte Carlo approximation. In our experiments, a value of α between 0.5 and 0.8 has proved to be a good compromise. We report in table 6.III the effect of ν on several problems, using a sample of 10000 pairs of tours, the first tour in a pair being obtained generating a random permutation of the cities, and second tour obtained after the applications of one of the moves reviewed in section 4.3, and denoting by $\bar{x}$ the empirical average of x. The table shows that in some cases, as in gr17, we could even take a value ν less than 1.

No	Problem	σ_i^2	σ_f^2	T_i	T_f	${T_f}^\eta$	ν
1	rnd8	12.25	0.12	10.5	0.45	0.41	**3**
2	bays29	12.25	0.03	40	1.66	1.74	**10**
3	gr17	12.25	0.39	45	2	2.14	**1**

Table 6.II: Estimating ν for CD with $\eta = 1.1$ and known σ_i^2

Lib	T_i	σ_i^2	$\overline{\lvert\Delta E\rvert}$	ν	$\overline{P_a}$
rnd8	10.5	12.25	10.99	3	**0.72**
bays29	40	12.25	256.04	10	**0.56**
gr17	45	12.25	313.07	1	**0.55**

Table 6.III: Empirical CD acceptance probability

6.2 Results of Experiments

We summarize the results of our experiments in table 6.IV, comparing some noise reduction strategies reviewed in section 3.2.1 over 200 SA replications. In the table, GP

stands for Gutjahr and Pflug and in brackets, we give the value used for the parameter γ. Similarly, we give the value of η in brackets for NSA. We report the best final tour length over the 200 replications as BFinal, as well as the best length over all SA iterations, identified as Best. The same quantities are reported when we introduce noise, denoting by BNFinal and NBest the best final length under noise and the overall best length under noise. We next give the average of the lengths over the 200 replications, and report the equivalent computation cost, normalizing the cost for one tour evaluation at the initial variance at one. The number of replications for which the last iteration corresponds to the optimal solution is given in the next column, and finally, we report in the last column the number of replications achieving 1%-optimality in the last iteration.

From the table, we can see that SA usually succeeds to stabilize on a 1%-optimal solution, but not necessarily on the optimal solution. Recording the best solution when noise is present produces a bias, the optimal tour length being underestimated, reflecting that under noise, it is safer to consider the last solutions only. We did not investigate other options to better select the solution to report. In all examples, NSA–CD outperforms the other methods in terms of solution quality and computation costs, the value $\eta = 1.2$ being a good compromise. We give more detailed comparisons in the next sections.

6.2.1 Rnd8

Since the initial tour obtained by the greedy algorithm has a distance of 93.81, as in figure 5.3 and table 5.IV, which is already optimal, we consider another initial tour, depicted in figure 6.1. The tour correspond the the coordinates $(5,10)$, $(15,5)$, $(30,5)$, $(20,15)$, $(25,20)$, $(30,30)$, $(20,18)$, $(10,20)$.

We compare in figure 6.2 NSA and GP techniques, with a temperature starting at 10.5 and decressing to 0.45. The curves correspond to the capability of the methods to find the optimal or 1%-optimal solution when the computational budget increases. The computation time of NSA–CD starts at 3 since ν equals 3. At the beginning, NSA–MH works better than the other methods but NSA–CD quickly dominates the other approaches. Both NSA–CD and GP–MH ultimately obtained a 1%-optimal solutions, while NSA–MH has a lower success rate, reflecting its theoretical weaknesses. The computational

No	$[T_i, T_f]$	Method	$[\sigma_i^2, \sigma_f^2]$	BFinal (Best)	BNFinal (NBest)	$\overline{\text{Final}}$ ($\overline{\text{Best}}$)	**$\overline{\text{Nfinal}}$ ($\overline{\text{NBest}}$)**	Comp. Cost	**Opt.**	1% opt.
	rnd8									
1	[10.5,0.45]	NSA(1.1)–MH	[12.25,0.38]	93.10 (93.10)	93.75 (89.71)	93.49 (93.42)	**93.52 (83.80)**	2.85e+08	**88**	176
2	[10.5,0.45]	GP(1.01)–MH	[12.25,7.93e-14]	93.10 (93.10)	93.10 (84.91)	93.37 (93.10)	**93.37 (83.8)**	5.22e+20	**112**	188
3	[10.5,0.45]	NSA(1.1,ν=3)–CD	[12.25,0.12]	93.10 (93.10)	92.48 (91.52)	93.31 (93.10)	**93.00 (91.13)**	9.45e+08	**128**	189
4	[10.5,0.45]	NSA(1.2,ν=3)–CD	[12.25,0.09]	93.10 (93.10)	92.68 (91.74)	93.31 (93.10)	**93.11 (91.25)**	1.28e+09	**123**	189
5	[10.5,0.45]	NSA(2,ν=3)–CD	[12.25,0.007]	93.10 (93.10)	93.14 (92.66)	93.41 (93.10)	**93.39 (92.24)**	1.35e+10	**109**	181
	gr17									
6	[45,2]	NSA(1.1)–MH	[12.25,0.39]	2085 (2085)	2084.81 (2081.61)	2085.36 (2085)	**2085.26 (2076.22)**	1.54e+08	**193**	200
7	[45,2]	GP(1.01)–MH	[12.25,2.54e-13]	2085 (2085)	2085 (2084.98)	2085.36 (2085)	**2085.36 (2076.38)**	9.44e+19	**193**	200
8	[45,2]	NSA(1.1)–CD	[12.25,0.39]	2085 (2085)	2086.34 (2082.66)	2085.38 (2085)	**2085.14 (2081.75)**	1.69e+08	**193**	200
9	[45,2]	NSA(1.2)–CD	[12.25,0.29]	2085 (2085)	2086.14 (2082.99)	2085.3 (2085)	**2085.11 (2082.12)**	2.29e+08	**194**	200
10	[45,2]	NSA(2)–CD	[12.25,0.02]	2085 (2085)	2085.33 (2084.39)	2085.3 (2085)	**2085.27 (2083.96)**	2.37e+09	**200**	200
	bays29									
11	[40,1.66]	NSA(1.1)–MH	[12.25,0.36]	2031 (2031)	2032.06 (2027.80)	2032.04 (2031.87)	**2032.1 (2023.37)**	1.85e+08	—	199
12	[40,1.66]	GP(1.01)–MH	[12.25,2.50e-13]	2031 (2031)	2031 (2031)	2031.55 (2031)	**2031.55 (2023.76)**	7.68e+19	—	199
13	[40,1.66]	NSA(1.1,ν=10)–CD	[12.25,0.03]	2031 (2031)	2030.98 (2030.46)	2031.63 (2031)	**2031.64 (2030.19)**	1.85e+09	—	200
14	[40,1.66]	NSA(1.2,ν=10)–CD	[12.25,0.02]	2031 (2031)	2030.99 (2030.54)	2031.65 (2031)	**2031.66 (2030.31)**	2.26e+09	—	200
15	[40,1.66]	NSA(2,ν=10)–CD	[1.225,0.002]	2031 (2031)	2031 (2030.87)	2031.9 (2031)	**2031.91 (2030.79)**	2.73e+10	—	200

Table 6.IV: Experimental results over 200 SA replications

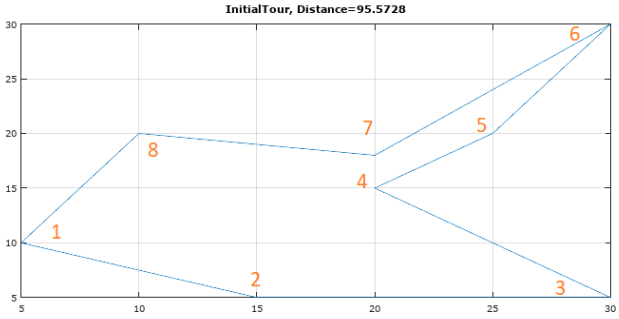


Figure 6.1: New initial tour for problem rnd8

effort required by GP–MH is nevertheless significantly higher than NSA–CD. We next investigate the choice of parameter η in figure 6.3. The computation cost increases along with η, as expected, while the solution quality seems to not depend on the parameter value.

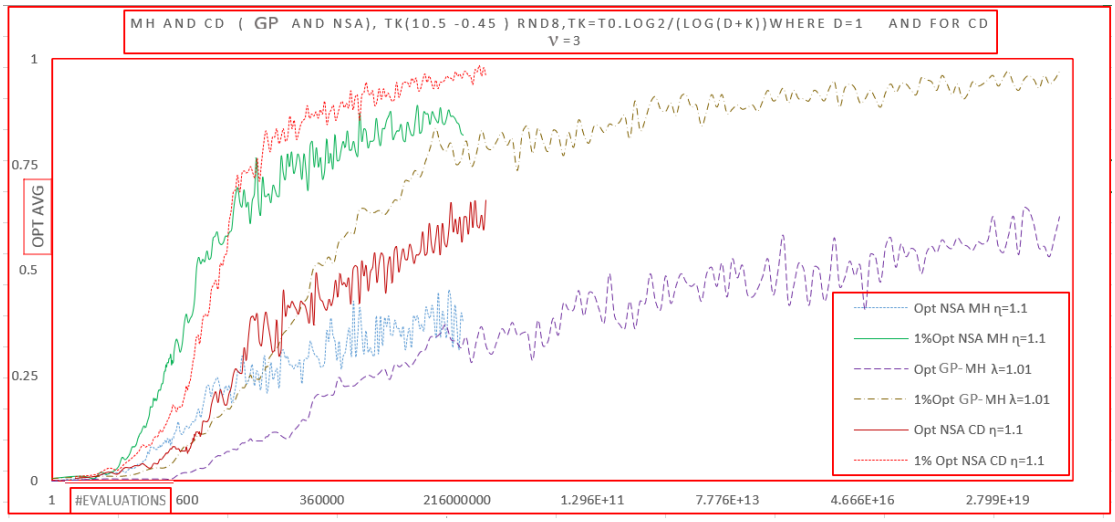


Figure 6.2: Comparison of SAUN methods for problem rnd8

6.2.2 Gr17

We report the same comparison between NSA and GP for problem gr17 in figure 6.4, for a temperature decreasing from 45 to 2. NSA–CD and NSA–MH perform in a similar way, ultimately reaching 1%-optimality on all the replications, and optimality on nearly

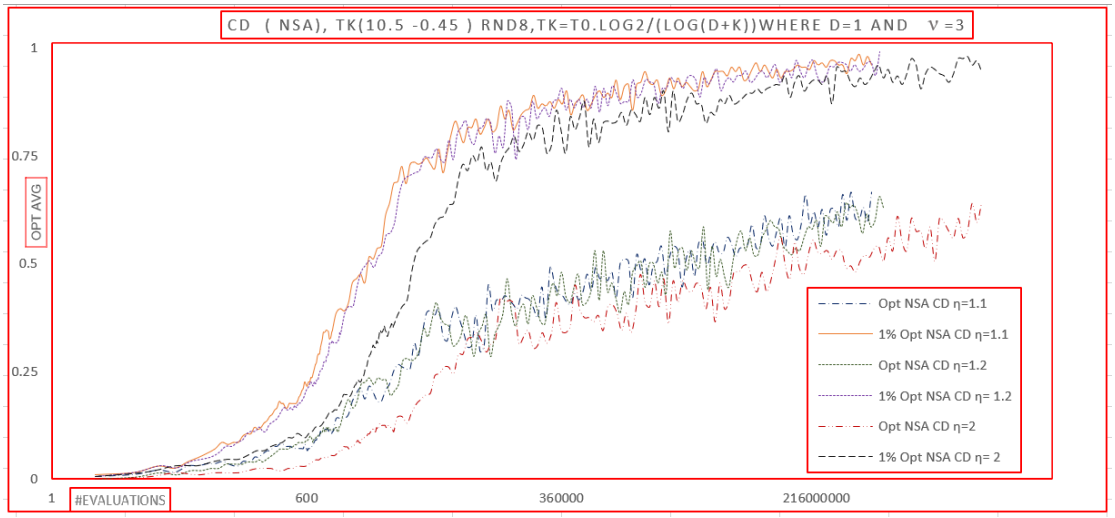


Figure 6.3: NSA–CD performance for problem rnd8

all the replications. Both approached clearly outperform GP–MH, that reached similar ratio of optimal and near-optimal solutions, but at the expense of a computation that increases exponentially faster. These observations are in line with the results reported in table 6.IV, exhibiting that the three methods are able to reach the optimal length 2085 on most of the SA replications, and 1%-optimal solution on all the simulations, but at a fraction of the cost for NSA compared to GP.

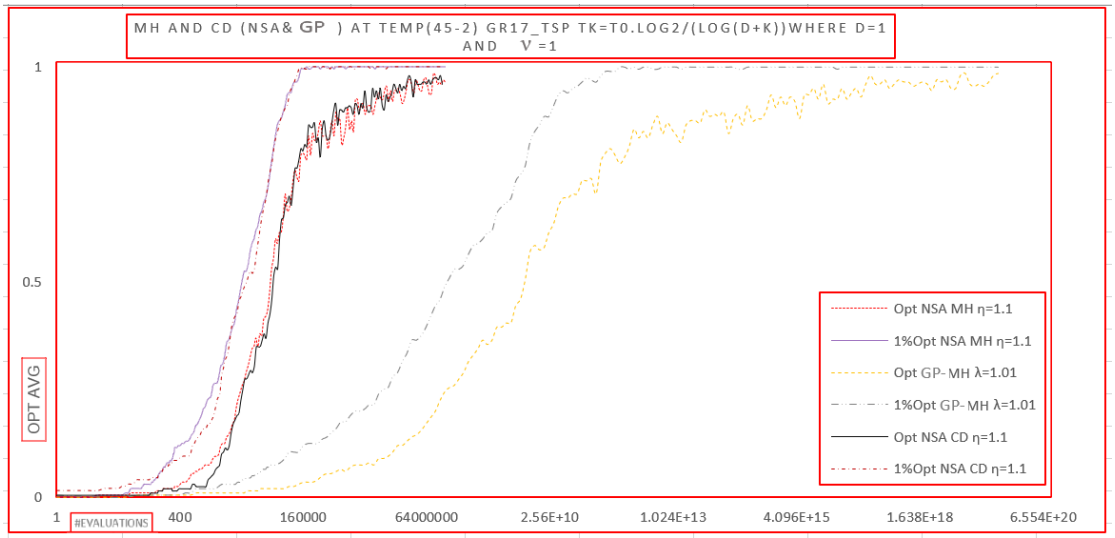


Figure 6.4: Comparison of SAUN methods for problem gr17

As in the previous example, figure 6.5 illustrates that NSA–CD achieves good performance even for a small value of η, while the computational cost decreases. The method is however slightly more robust with $\eta = 2$ as it reaches 1%-optimality on all SA replications, while near optimality is achieved on 96.5% and 97% of the replications for $\eta = 1.1$ and $\eta = 1.2$, respectively. We can observe in table 6.IV that GP achieves near-optimality on all the replications, but the required computation cost is prohibitive.

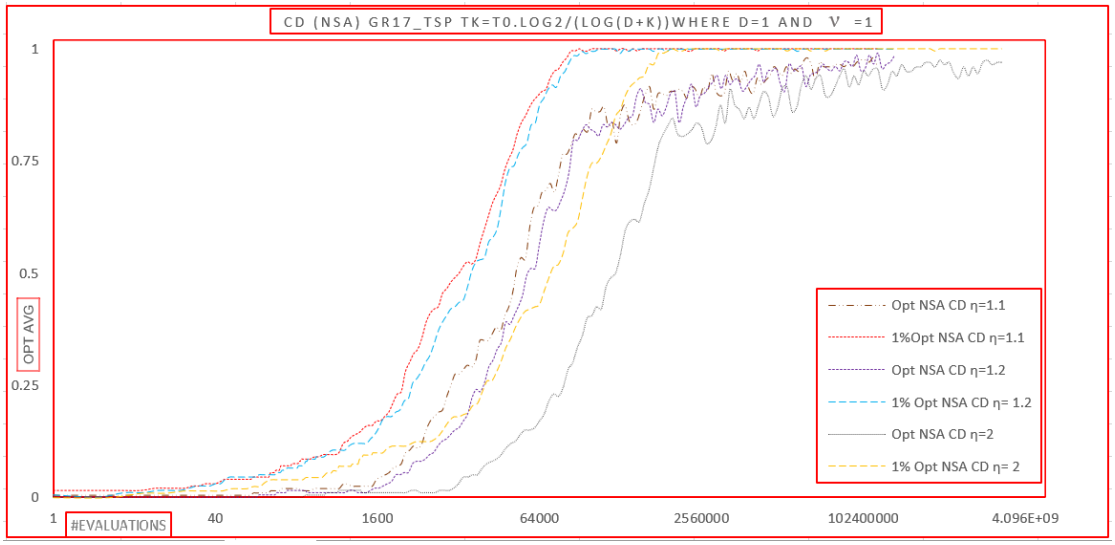


Figure 6.5: NSA–CD performance for problem gr17

6.2.3 Bays29

We finally report results for problem bays29, for which SA had issues to stabilize in the optimal solution, as no method managed to produce the optimal tour in the last iteration, as reported in table 6.IV. We therefore only compare 1%-optimality in figure 6.6, decreasing the temperature from 40 to 1.66. While we had to reduce the initial variance by 10 for NSA–CD, it exhibits a sharp increase in the ratio of 1%-optimal solutions found with the the computational budget, and manages to attain a good solution in nearly all the replications, as shown again in table 6.IV. We did not apply such an initial variance reduction for NSA–MH, which achieves the same success rate than NSA–CD, but with a lower cost. As in the previous problems, GP ultimately exhibits the same

success rate, but at the price of a much higher computation cost, making the approach not competitive.

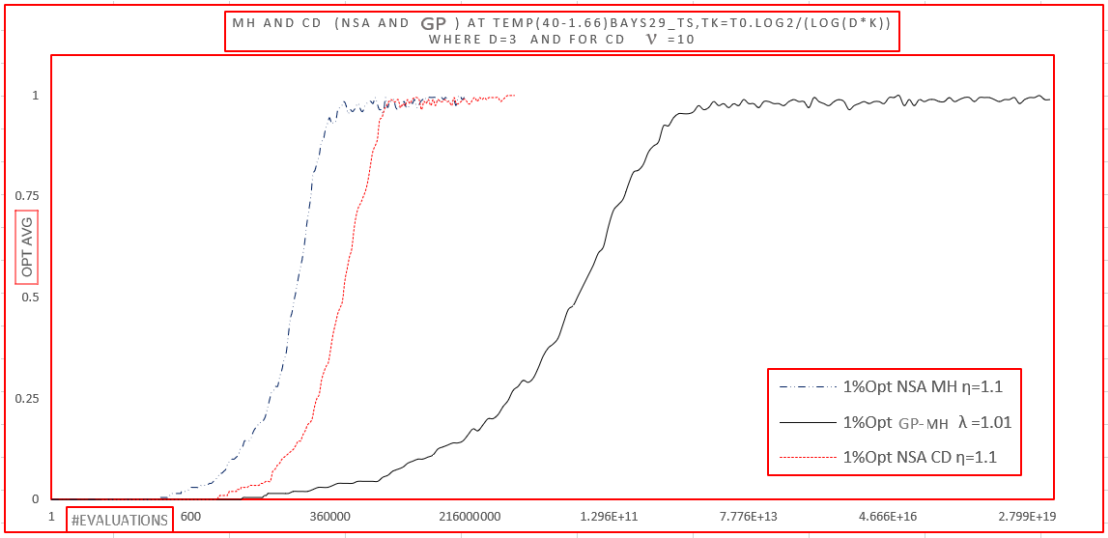


Figure 6.6: Comparison of SAUN methods for problem bays29

6.3 Discussion

The experiments show that SA usually needs many iterations before we can reach convergence, and we have to carefully choose initial and final temperatures in order to diversify the search during the first iterations, while allowing to converge to a good solution during the final iterations. In presence of noise, Gutjahr and Pflug's method allows to find optimal or nearly-optimal solutions, but a prohibitive cost, while NSA with the acceptance technique proposed by Ceperley and Dewing succeeds to discover the solutions with much less computational efforts. NSA with Metropolis-Hastings acceptance often delivers good numerical results, but in one problem, it underperformed in terms of solution quality. This is no surprising as there is no a theoretical guarantee for such a combination. Therefore, the best results were obtained with NSA–CD, but in our experimations, we have discovered that the method can be quite sensitive to the choice of initial and final temperatures, and the value of the initial variance, that has to be limited. It nevertheless appears from our results that SAUN methods are promising to tackle

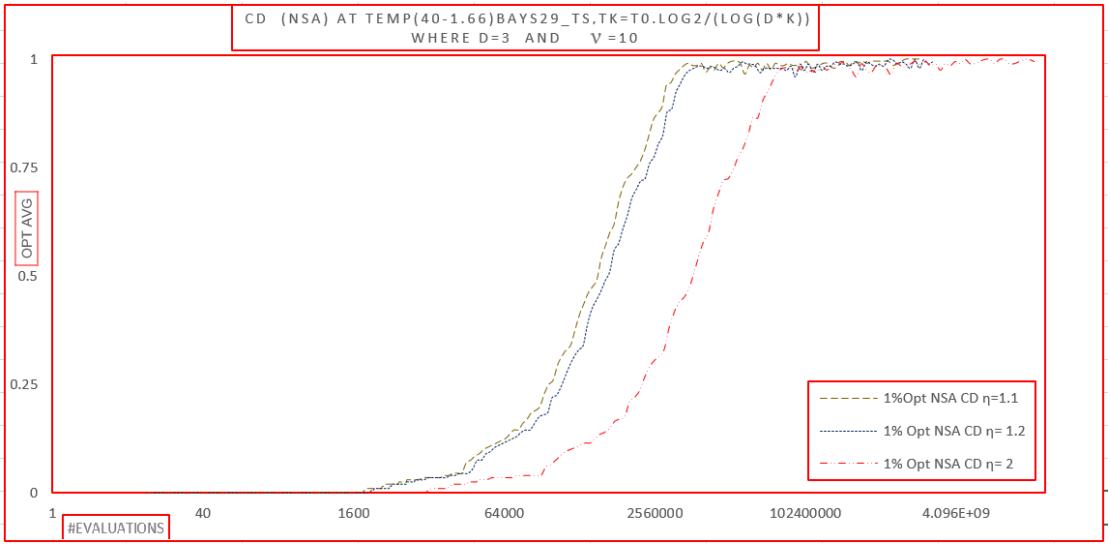


Figure 6.7: NSA–CD performance for problem bays29

problems under noise when only metaheuristics can be used.

CHAPTER 7

CONCLUSION

In this thesis, we explore the use of simulated annealing for optimization problems affected by noise. Previous studies exhibit that convergence can still be obtained, but the computational cost exponentially increases as the temperature goes to zero [31]. Our experiments on the traveling salesman problem (TSP) are in line with these observations, but the efficiency of the simulated annealing can be greatly improved. The acceptance criterion proposed by Ceperley and Dewing [16] (CD) indeed provides important computational savings if we decrease the variance along with the temperature, a point previously ignored [15]. The variance reduction ratio can be kept close to proportional to the temperature decrease, an approach considered in the proposed Noisy Simulated Annealing (NSA), while other strategies taking noise under consideration typically impose the variance to be in the order of the square of the temperature [31]. As a result, NSA with CD achieves a much faster convergence rate, but the numerical experiments suggest that the method needs a low final temperature in order to outperform the other approaches. NSA with the standard Metropolis-Hastings acceptance criterion gave sometimes better results than expected, being competitive with the best techniques, but was the less robust method on other TSP instances, reflecting the lack of convergence guarantees in presence of noise.

Simulated annealing however suffers from several limitations. The cost function used to model problems has to be simple and its evaluation, fast. The solution space should no be restricted, but it is possible to limit the neighborhood used to determine the candidate solution at a given iteration. In presence of noise, simulated annealing under noise (SAUN) strategies outperform the classical simulated annealing method. The efficiency of SAUN crucially depends on the decrease of the randomness in the problem, that can be performed in different ways. The decrease speed has to be controlled and if possible, reduced. NSA achieves this objective.

While promising, the numerical experiments remain limited and should be extended.

We first could explore use of larger temperatures value and limit final iterations where nearly no progress is observed in the solution and the variance does not decrease significantly as the temperature cooling is very slow. Second, more examples, if not all, from TSPLIB should be analyzed. Ideally, we could create a noisy version of TSPLIB, along with the best solutions, computation costs, and noise levels in order to compare with any technique proposed to handle noisy problems. We could consider various noise distributions, such as rectangular distribution, triangular distribution, Maxwell distribution, etc., as suggested by Gutjahr and Pflug [31] in their conclusion. Providing many examples however needs more time, especially for the largest instances and more expensive, especially for time consuming approaches as the Gutjahr-Pflug method. On the other hand, the performance of NSA with CD was sensitive to the initial variance level, so more attention should be devoted to this point. We could consider a general method for setting the initial variance, similar to what we have done for NSA with CD, with the hope to limit the computational cost during the final iterations due to some compromise between the final temperature and variance. Moreover, in order to keep the problems numerically manageable, we set the variance value when drawing the error in our experiments, rather than keeping the variance fixed, and averaging the observations over n experiments, as in real applications.

Finally, while Gutjahr and Pflug [31] formally prove that their approach asymptotically converges towards the set of global minimizers of the objective function under consideration, a theoretical proof of the convergence of NSA with CD has still to be provided, as Ceperley and Dewing [16] only consider the case of a fixed temperature. The proof could follow similar lines than convergence of the classical SA [1], and could give more insight of the method parameters as well as the temperature decrease scheme.

BIBLIOGRAPHY

[1] E. H. L. Aarts and J. H. M. Korst. *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing*. John Wiley & Sons, New York, NY, USA, 1989.

[2] E. H. L. Aarts and P. J. M. van Laarhoven. Simulated annealing: an introduction. *Statistica Neerlandica*, 43(1):31–52, 1989.

[3] E. H. L. Aarts, J. H. M. Korst, and W. Michiels. Simulated annealing. In E. K. Burke and G. Kendall, editors, *Search methodologies*, pages 265–285. Springer, 2014.

[4] H. Aguiar e Oliveira Jr, L. Ingber, A. Petraglia, M. R. Petraglia, and M. A. S. Machado. *Stochastic global optimization and its applications with fuzzy adaptive simulated annealing*. Springer-Verlag, Berlin Heidelberg, Germany, 2012.

[5] T. M. Alkhamis, M. A. Ahmed, and V. K. Tuan. Simulated annealing for discrete optimization with estimation. *European Journal of Operational Research*, 116(3): 530–544, 1999.

[6] M. H. Alrefaei and S. Andradóttir. A simulated annealing algorithm with constant temperature for discrete stochastic optimization. *Management Science*, 45(5):748–764, 1999.

[7] S. Amaran, N. V. Sahinidis, B. Sharda, and S. J. Bury. Simulation optimization: a review of algorithms and applications. *Annals of Operations Research*, 240(1): 351–380, 2016.

[8] S. Andradóttir. A review of random search methods. In M. C. Fu, editor, *Handbook of Simulation Optimization*, pages 277–292. Springer, New York, NY, USA, 2015.

[9] S. Anily and A. Federgruen. Simulated annealing methods with general acceptance probabilities. *Journal of Applied Probability*, 24(3):657–667, 1987.

[10] J. Basel and T. R. Willemain. Random tours in the traveling salesman problem: analysis and application. *Computational Optimization and Applications*, 20(2): 211–217, 2001.

[11] W. Ben-Ameur. Computing the initial temperature of simulated annealing. *Computational Optimization and Applications*, 29(3):369–385, 2004.

[12] D. Bertsimas and J. Tsitsiklis. Simulated annealing. *Statistical Science*, 8(1):10–15, 1993.

[13] C. Bouttier and I. Gavra. Convergence rate of a simulated annealing algorithm with noisy observations. *arXiv preprint arXiv:1703.00329*, 2017.

[14] N. E. Bowler, T. M. A. Fink, and R. C. Ball. Characterization of the probabilistic traveling salesman problem. *Physical Review E*, 68(3):036703, 2003.

[15] J. Branke, S. Meisel, and C. Schmidt. Simulated annealing in the presence of noise. *Journal of Heuristics*, 14(6):627–654, 2008.

[16] D. M. Ceperley and M. Dewing. The penalty method for random walks with uncertain energies. *The Journal of chemical physics*, 110(20):9812–9820, 1999.

[17] V. Černỳ. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of optimization theory and applications*, 45 (1):41–51, 1985.

[18] A. Charnes and M. Wolfe. Extended Pincus theorems and convergence of simulated annealing. *International Journal of Systems Science*, 20(8):1521–1533, 1989.

[19] E. A. B. Cole. *Mathematical and Numerical Modelling of Heterostructure Semiconductor Devices: From Theory to Programming*, chapter Genetic algorithms and simulated annealing, pages 339–376. Springer, 2009.

[20] J. R. Cruz and C. C. Y. Dorea. Simple conditions for the convergence of simulated annealing type algorithms. *Journal of applied probability*, 35(4):885–892, 1998.

[21] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, 2002.

[22] M. Duque-Antón. Constructing efficient simulated annealing algorithms. *Discrete Applied Mathematics*, 77(2):139–159, 1997.

[23] R. W. Eglese. Simulated annealing: a tool for operational research. *European journal of operational research*, 46(3):271–281, 1990.

[24] V. Fabian. Simulated annealing simulated. *Computers & Mathematics with Applications*, 33(1-2):81–94, 1997.

[25] T. M. Fink. *Inverse protein folding, hierarchical optimisation and tie knots*. PhD thesis, University of Cambridge, 1998.

[26] D. Fouskakis and D. Draper. Stochastic optimization: a review. *International Statistical Review*, 70(3):315–349, 2002.

[27] S. B. Gelfand and S. K. Mitter. Simulated annealing with noisy or imprecise energy measurements. *Journal of Optimization Theory and Applications*, 62(1):49–62, 1989.

[28] G. H. Givens and J. A. Hoeting. *Computational statistics*. John Wiley & Sons, Hoboken, NJ, USA, 2 edition, 2012.

[29] R. J. Glauber. Time dependent statistics of the Ising model. *Journal of Mathematical Physics*, 4(2):294–307, 1963.

[30] W. J. Gutjahr. Recent trends in metaheuristics for stochastic combinatorial optimization. *Central European Journal of Computer Science*, 1(1):58–66, 2011.

[31] W. J. Gutjahr and G. C. Pflug. Simulated annealing for noisy cost functions. *Journal of Global Optimization*, 8(1):1–13, 1996.

[32] B. Hajek. Cooling schedules for optimal annealing. *Mathematics of operations research*, 13(2):311–329, 1988.

[33] A. Henchiri, M. Bellalouna, and W. Khaznaji. A probabilistic traveling salesman problem: a survey. In *Position papers of the 2014 Federated Conference on Computer Science and Information Systems*, volume 3, pages 55–60. Annals of Computer Science and Information Systems, 2014.

[34] D. Henderson, S. H. Jacobson, and A. W. Johnson. The theory and practice of simulated annealing. In F. Glover and G. A. Kochenberger, editors, *Handbook of metaheuristics*, pages 287–319. Springer, Boston, MA, USA, 2003.

[35] L. Ingber. Simulated annealing: Practice versus theory. *Mathematical and computer modelling*, 18(11):29–57, 1993.

[36] D. S. Johnson and L. A. McGeoch. The traveling salesman problem: A case study in local optimization. In E. H. L. Aarts and J. K. Lenstra, editors, *Local Search in Combinatorial Optimization*, pages 215–310. John Wiley and Sons, Chichester, United Kingdom, 1997.

[37] D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon. Optimization by simulated annealing: an experimental evaluation; part I, graph partitioning. *Operations research*, 37(6):865–892, 1989.

[38] D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon. Optimization by simulated annealing: an experimental evaluation; part II, graph coloring and number partitioning. *Operations research*, 39(3):378–406, 1991.

[39] M. Jünger, G. Reinelt, and G. Rinaldi. The traveling salesman problem. In M. Ball, T. Magnanti, C. Monma, and G. Nemhauser, editors, *Network Models*, volume 7 of *Handbooks in Operations Research and Management Science*, pages 225–330. Elsevier, 1995.

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

[41] P. J. M. van Laarhoven and E. H. L. Aarts. *Simulated Annealing: Theory and Applications*. Kluwer, Dordrecht, The Netherlands, 1987.

[42] P. L'Ecuyer. Good parameters and implementations for combined multiple recursive random number generators. *Operations Research*, 47(1):159–164, 1999.

[43] S. Ledesma, G. Aviña, and R. Sanchez. Practical considerations for simulated annealing implementation. In C. M. Tan, editor, *Simulated Annealing*, volume 20, pages 401–420. InTech, Vienna, Austria, 2008.

[44] S. Lin and B. W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21(2):498–516, 1973.

[45] A. Lodi and A. P. Punnen. TSP software. In G. Gutin and A. P. Punnen, editors, *The traveling salesman problem and its variations*, volume 12 of *Combinatorial Optimization*, pages 737–749. Springer, Boston, MA, USA, 2006.

[46] T. H. de Mello and G. Bayraksan. Stochastic constraints and variance reduction techniques. In M. C. Fu, editor, *Handbook of Simulation Optimization*, pages 245–276. Springer, 2010.

[47] A. G. Nikolaev and S. H. Jacobson. Simulated annealing. In M. Gendreau and J.-Y. Potvin, editors, *Handbook of Metaheuristics*, pages 1–39. Springer, Boston, MA, USA, 2010.

[48] Y. Nourani and B. Andresen. A comparison of simulated annealing cooling strategies. *Journal of Physics A: Mathematical and General*, 31(41):8373–8385, 1998.

[49] R. H. J. M. Otten and L. P. P. P van Ginneken. *The annealing algorithm*. Kluwer, Dordrecht, The Netherlands, 1989.

[50] C. H. Papadimitriou. The Euclidean travelling salesman problem is NP-complete. *Theoretical computer science*, 4(3):237–244, 1977.

[51] D. T. Pham and D. Karaboga. *Intelligent Optimisation Techniques: Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks*. Springer-Verlag, London, United Kingdom, 2000.

[52] E. Platen and N. Bruti-Liberati. *Numerical Solution of Stochastic Differential Equations with Jumps in Finance*, chapter Variance Reduction Techniques, pages 637–695. Springer-Verlag, Berlin Heidelberg, Germany, 2010.

[53] A. P. Punnen. The traveling salesman problem: Applications, formulations and variations. In G. Gutin and A. P. Punnen, editors, *The traveling salesman problem and its variations*, pages 1–28. Springer, Boston, MA, USA, 2007.

[54] C. Rego and F. Glover. Local search and metaheuristics. In G. Gutin and A. P. Punnen, editors, *The traveling salesman problem and its variations*, pages 309–368. Springer, Boston, MA, USA, 2006.

[55] G. Reinelt. TSPLIB–a traveling salesman problem library. *ORSA Journal on Computing*, 3(4):376–384, 1991.

[56] G. Reinelt. *The traveling salesman: computational solutions for TSP applications*. Springer-Verlag, Berlin Heidelberg, Germany, 1994.

[57] S. L. Rosen and C. M. Harmonosky. An improved simulated annealing simulation optimization method for discrete parameter stochastic systems. *Computers & Operations Research*, 32(2):343–358, 2005.

[58] R. A. Rutenbar. Simulated annealing algorithms: an overview. *IEEE Circuits and Devices Magazine*, 5(1):19–26, 1989.

[59] S. M. Sait and H. Youssef. *VLSI physical design automation: theory and practice*. World Scientific, Singapore, 1999.

[60] S. M. Saıt and H. Youssef. *Iterative computer algorithms with applications in engineering*. IEEE Computer Society Press, Washington, D.C., USA, 1999.

[61] J. Schneider and S. Kirkpatrick. *Stochastic optimization*. Springer-Verlag, Berlin Heidelberg, Germany, 2006.

[62] Alexander Shapiro, Darinka Dentcheva, and Andrzej Ruszczyński. *Lectures on Stochastic Programming*. SIAM, Philadelphia, PA, USA, 2009.

[63] J. C. Spall. *Introduction to stochastic search and optimization: estimation, simulation, and control*. John Wiley & Sons, Hoboken. NJ, USA, 2003.

[64] C. M. Tan and N. Raghavan. Simulated annealing for mixture distribution analysis and its applications to reliability testing. In C. M. Tan, editor, *Simulated Annealing*. InTech, Vienna, Austria, 2008.

[65] J. M. Varanelli. *On the acceleration of simulated annealing*. PhD thesis, University of Virginia, 1996.

[66] R. V. V. Vidal, editor. *Applied simulated annealing*, volume 396 of *Lectures Notes in Economics and Mathematical Systems*. Springer-Verlag, Berlin Heidelberg, Germany, 1993.

[67] D. Weyland. Simulated annealing, its parameter settings and the longest common subsequence problem. In *Proceedings of the 10th annual conference on genetic and evolutionary computation*, pages 803–810. ACM, 2008.

[68] X. Yao and G. Li. General simulated annealing. *Journal of Computer Science and Technology*, 6(4):329–338, 1991.

[69] Z. B. Zabinsky. Stochastic methods for practical global optimization. *Journal of Global Optimization*, 13(4):433–444, 1998.

[70] Z. B. Zabinsky. Random search algorithms. In J. J. Cochran, L. A. Cox, P. Keskinocak, J. P. Kharoufeh, and J. C. Smith, editors, *Wiley Encyclopedia of Operations Research and Management Science*. John Wiley & Sons, Hoboken, NJ, USA, 2010.

[71] A. Zhigljavsky and A. Žilinskas. *Stochastic Global Optimization*. Springer, Boston, MA, USA, 2008.