SIMULATED ANNEALING
Strategies, Potential Uses and Advantages

Marcos de Sales Guerra Tsuzuki
Thiago de Castro Martins
Editors
SIMULATED ANNEALING

STRATEGIES, POTENTIAL USES AND ADVANTAGES
MATHEMATICS RESEARCH DEVELOPMENTS

Additional books in this series can be found on Nova’s website under the Series tab.

Additional e-books in this series can be found on Nova’s website under the e-book tab.

Complimentary Contributor Copy
SIMULATED ANNEALING

STRATEGIES, POTENTIAL USES AND ADVANTAGES

MARCOS DE SALES GUERRA TSUZUKI
AND
THIAGO DE CASTRO MARTINS
EDITORS

Complimentary Contributor Copy
# CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preface</td>
<td></td>
<td>vii</td>
</tr>
<tr>
<td></td>
<td>About the Editors</td>
<td>ix</td>
</tr>
<tr>
<td>Chapter 1</td>
<td>Simulated Annealing: In Mathematical Global Optimization Computation, Hybrid with Local or Global Search, and Practical Applications in Crystallography and Molecular Modelling of Prion Amyloid Fibrils</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Jiapu Zhang</td>
<td></td>
</tr>
<tr>
<td>Chapter 2</td>
<td>Handling Equality and Inequality Linear Constraints in Simulated Annealing</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>Danilo R. Velis</td>
<td></td>
</tr>
<tr>
<td>Chapter 3</td>
<td>Aircraft Design with Simulated Annealing for Sae Aerodesign Competition</td>
<td>55</td>
</tr>
<tr>
<td></td>
<td>Diego Movio Borburema, Marcos de Sales Guerra Tsuzuki, Antonio Luis de Campos Mariani, Andre Kubagawa Sato and Thiago de Castro Martins</td>
<td></td>
</tr>
<tr>
<td>Chapter 4</td>
<td>Simulated Annealing Applied to the Resolution of Phylogenetic Reconstruction with Maximum Parsimony</td>
<td>71</td>
</tr>
<tr>
<td></td>
<td>Jean-Michel Richer, Karla E. Vazquez-Ortiz and David Lesaint</td>
<td></td>
</tr>
</tbody>
</table>
## Contents

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 5</td>
<td>Improved Parallel Simulated Annealing Applied to Spacecraft Trajectory Optimization</td>
<td>99</td>
</tr>
<tr>
<td></td>
<td>Ya-Zhong Luo and Guo-Jin Tang</td>
<td></td>
</tr>
<tr>
<td>Chapter 6</td>
<td>Simulated Annealing Applications in Thermodynamic Calculations</td>
<td>139</td>
</tr>
<tr>
<td></td>
<td>Ahmed O. Elnabawy, Seif-Eddeen K. Fateen and Adrian Bonilla-Petriciolet</td>
<td></td>
</tr>
<tr>
<td>Chapter 7</td>
<td>SA Optimization of Two-Chamber Mufflers Equipped with Multiple Parallel Perforated Tubes within a Constrained Space</td>
<td>159</td>
</tr>
<tr>
<td></td>
<td>Min-Chie Chiu</td>
<td></td>
</tr>
<tr>
<td>Chapter 8</td>
<td>A Simulated Annealing Algorithm to Solve the Log-Truck Scheduling Problem</td>
<td>191</td>
</tr>
<tr>
<td></td>
<td>Mauricio Acuna and John Sessions</td>
<td></td>
</tr>
<tr>
<td>Chapter 9</td>
<td>Potential Applications of Simulated Annealing to Multi-Species Fish Habitat Conservation in the Gulf of Maine</td>
<td>221</td>
</tr>
<tr>
<td></td>
<td>Rosamonde R. Cook and Peter J. Auster</td>
<td></td>
</tr>
<tr>
<td>Index</td>
<td></td>
<td>243</td>
</tr>
</tbody>
</table>

Complimentary Contributor Copy
Preface

Simulated Annealing (SA) is a generic probabilistic meta-heuristic that searches the global optimum in a large search space. The search space can have discrete and continuous variables and the objective function can be discrete or continuous. The optimization problem can be constrained or not. This book aims to present SA key points and several applications.

In Chapter 1, a detailed explanation about SA key components is performed (initial temperature, cooling schedule, neighborhood function and others). SA is used to solve several problems from the literature. Chapter 2 shows a technique in which linear constraints are transformed into domain boundaries. The original variables are expressed in terms of a new set of variables through the singular value decomposition of the constraint matrix. Hence, feasibility is kept at every SA stage, avoiding unnecessary objective function evaluation due to constraints violations.

A new neighborhood function that determines the next candidate (called crystallization factor) is shown in Chapter 3. It can be used with discrete and continuous objective functions with continuous variables. The application to aircraft design simulation optimization is shown. In Chapter 4 it is proposed a new SA called SAMPARS (Simulated Annealing for Maximum PARSimony). Different possibilities for the AS key components (initial temperature, cooling schedule, improved neighborhood function) were carefully analyzed and tuned. Its application to reconstruct phylogenetic tree from DNA sequences by minimizing the number of genetic transformations is investigated.

A hybridized SA with simplex is shown in Chapter 5. SA and simplex are complementary methods, in the sense that global and local searches can be performed. The proposed method is applied to optimize spacecraft trajectory that is transferring the orbit from Earth to Mars. Two performance indexes are
considered: minimum fuel consumption and minimum transfer time. Chapter 6 shows SA applications in thermodynamics. SA has a close connection with thermodynamics, and several thermodynamics calculations are performed using SA: phase equilibrium, chemical equilibrium, simultaneous phase and chemical equilibrium, phase stability, calculation of critical points and azeotropes, among others.

Chapter 7 shows an application in sound transmission loss. Recently, hybridized mufflers with single perforated tube have been researched. Multiple parallel perforated tubes have the potential of dispersing the venting fluid and reduce secondary flow noise. Sound transmission loss of two chamber mufflers equipped with parallel perforated tubes is optimized and the best design shape within a limited space is determined. Chapter 8 solve the log-truck scheduling problem, this is an extension of the timber transport vehicle routing problem with time windows. The problem is characterized by a heterogeneous fleet of trucks that depart from their corresponding depots and must perform a number of daily transport tasks to move log products between wood pick up (harvest areas) locations and industrial customers, such as pulp mills and sawmills.

Chapter 9 designs scenarios for marine protected areas intended to satisfy the Essential Fish Habitat requirements of the US Sustainable Fisheries Act for 15 economically valuable demersal fish species in the Gulf of Maine on the US Eastern Continental Shelf. Standardized fisheries independent surveys data from a long term monitoring program are used to test the algorithm.
ABOUT THE EDITORS

Dr. Marcos de Sales Guerra Tsuzuki was born in São Paulo, Brazil, in 1963. He received his electrical engineer degree in 1985 from the Engineering School from University of São Paulo, São Paulo, Brazil, and a Master's degree in engineering in 1989 from Yokohama National University, Yokohama, Japan. In 1995, he received his Ph.D. degree from the Engineering School of University of São Paulo. He is an Associate Professor at the Department of Mechatronics and Mechanical Systems Engineering, University of São Paulo since 2000, where he teaches courses on CAD/CAM, computer graphics, image processing and medical imaging. His current research areas include solid modeling, geometric modeling, bioengineering and biomedical inverse problems.

Dr. Thiago de Castro Martins was born in São Paulo, Brazil, in 1979. He received his mechanical engineer degree in 2001 from the Engineering School of the University of São Paulo, São Paulo, Brazil. In 2007, he received his Ph.D. degree from the Engineering School of University of São Paulo. He has been Assistant Professor in the Department of Mechatronics and Mechanical Systems Engineering, University of São Paulo since 2010, where he teaches courses on Electronics, Mechanical Design, computer graphics, image processing and medical imaging. His current research areas include combinatorial problems, nonlinear optimization, bioengineering and biomedical inverse problems.
Chapter 1

SIMULATED ANNEALING: IN MATHEMATICAL GLOBAL OPTIMIZATION COMPUTATION, HYBRID WITH LOCAL OR GLOBAL SEARCH, AND PRACTICAL APPLICATIONS IN CRYSTALLOGRAPHY AND MOLECULAR MODELLING OF PRION AMYLOID FIBRILS

Jiapu Zhang*
School of Science, Informatics Technology, Engineering and Centre of Informatics and Applied Optimisation, The University of Ballarat, MT Helen Campus, Victoria, Australia

ABSTRACT

Simulated annealing (SA) was inspired from annealing in metallurgy, a technique involving heating and controlled cooling of a material to increase the size of its crystals and reduce their defects, both are attributes of the material that depend on its thermodynamic free energy. In this Chapter, firstly we will study SA in details on its initial feasible solution choosing, initial temperature selecting, neighbourhood solution searching,}

*Tel: 61-423487360; 61-353276335, Email: jiapu_zhang@hotmail.com, j.zhang@ballarat.edu.au.

Complimentary Contributor Copy
efficient way of calculating for the difference of objective function values of two neighbourhod solutions, acceptance function (Metropolis function), temperature cooling, and the criteria of inner and outer loops' stopping, etc. Then, hybrid pure SA with local (or global) search optimization methods allows us to be able to design several effective and efficient global search optimization methods. In order to keep the original sense of SA, we clarify our understandings of SA in crystallography and molecular modelling field through the studies of prion amyloid fibrils.

**Keywords:** Simulated Annealing; Computational Algorithm; Global Optimization Search; Hybrid with Local Search; Applications

### 1. INTRODUCTION

Simulated Annealing (SA), as well as Tabu Search and Genetic Algorithm, is one of the successful heuristic computational methods. It simulates the annealing process with Monte Carlo property. The works of Metropolis, Kirkpatrick, Johnson, Aarts, et al., are well-known. In discrete optimization, simulated annealing method has found a lot of applications. The book for example [1] is good collections of its applications to discrete optimization problems. For continuous optimization problems, there are a lot of references on it. However, we still rarely see a very successful simulated annealing method for large scale continuous optimization problems in very high dimensions, especially in the constrained case.

The SA method appeared as early as in 1953 [2] as a Monte Carlo method and was firstly investigated and used in 1983 by Kirkpatrick et al. [3]. SA is a stochastic method. It differs from the traditional descent methods (see, for example, [4] and references therein) in that a local search algorithm for a neighborhood solution search, whether it randomly descents or steeply descents, allows downhill moves only, while in an attempt to escape local optima SA algorithm allows occasional uphill moves as well. SA techniques are based upon the physical analogy of cooling crystal structures (including the case of so called quenching) that spontaneously arrives at a stable configuration, characterized by-globally or locally-minimal potential energy. Starting with an initial solution x, and an initial “temperature”$T_0$, which is a parameter, we obtain a neighboring solution $x'$ and compare its cost with that of x. If the cost of $x'$ is smaller than that of x, i.e., $f(x') < f(x)$, we accept the new solution $x'$. The same thing would happen if we are applying the local search method by random descent method [4]. On the other hand, if $f(x')$ is
greater than \( f(x) \), (in which case local search algorithms (see, for example, [4]) will not accept \( x' \)), the SA algorithm may accept \( x' \), but with a probability \( e^{-\Delta x' / T_0} \), where \( \Delta x' \) is the difference in the costs of \( x' \) and \( x \). i.e., \( \Delta x' = f(x') - f(x) \). This process is carried out for a certain number of times, which we call iterations, for each temperature. Then we reduce the temperature according to a particular schedule, and repeat. The convergence of SA algorithms are studied, for example, in [5-6].

An essential element of the SA algorithm is the probability \( e^{-\Delta x' / T} \) of an uphill move of size \( \Delta x' \) being accepted when the current temperature is \( T \). This is dependent on both \( \Delta x' \) and \( T \). For a fixed temperature \( T \), smaller uphill moves \( \Delta x' \) have a higher probability of being accepted. On the other hand, for a particular uphill move \( \Delta x' \), a higher temperature will result in a larger probability for that uphill move being accepted. As stated in [7], “at a high temperature any uphill move might be indiscriminately accepted with large probability so that the objective function and the tumbles around the space are not very important; as \( T \) is lowered the objective function becomes more and more significant; until as \( T \) goes to zero the search becomes trapped in the lowest minima that it has reached.”

The SA algorithm for solving a practical problem is typically implemented in two nested loops: the outer loop and the inner loop. The outer loop controls temperatures, while the inner loop iterates a fixed number of times for the given temperature. The inner loop is for the problem specific decisions. The decisions of the outer loop involve the setting of initial temperature \( (T_0) \), the cooling schedule, the temperature length which is the number of outer loop iterations performed at each temperature, and the stopping condition of the outer loop. The inner loop of SA typically considers the following aspects: feasible solution space, initial feasible solution, neighborhood move, objective function values (and efficient calculation of their difference), and the decision which decides whether the move is found acceptable or probably acceptable according the so-called Metropolis criterion.

In this Chapter, Section 2 will study SA in details on its initial feasible solution choosing, initial temperature selecting, neighborhood solution searching, efficient way of calculating for the difference of objective function values of two neighborhood solutions, acceptance function (Metropolis function), temperature cooling, and the criteria of inner and outer loops’ stopping, etc. In Section 3, hybrid pure SA with local (or global) search optimization methods allows us to be able to design several effective and efficient global search optimization methods. In order to keep the original sense of SA, we clarify our understandings of SA in crystallography and
molecular modeling field through the studies of prion amyloid fibrils in Section 4. Section 5 will give some concluding remarks on SA.

2. IMPLEMENTING THE SA ALGORITHM

2.1. Overview

In this section we consider problem

\[
\text{Minimize } f(x) \text{ subject to } x \in X,
\]

where \( X \) is a subset of \( \mathbb{R}^n \) is a compact set and \( f \) is continuous, being solved by SA algorithm.

The word renew denotes the counts of the solution being accepted in the inner loop. The pseudo-code (referred, for example, to [8]) of the SA algorithm is listed as follows:

Algorithm 1. Simulated annealing algorithm.

**Initialization:**
- Define the objective function \( f \) and its feasible solution space.
- Call **Initial feasible solution producing procedure** to produce an initial feasible solution \( x \).
- Call **Procedure of selecting initial temperature** to produce the initial temperature \( T_0 \).
- Calculate the size of neighborhood \( N_{size} \).
- Calculate \( f(x) \), and set \( x_{\text{best}} = x \) and \( f_{\text{best}} = f(x) \).
- Set \( \text{best\_count} = \text{frozen\_count} = 0 \), and value of \( \delta \).

**Cooling (outer loop procedure):**
- Repeat (outer loop)
  - Call **Inner loop procedure**.
  - Call **cooling schedule** \( T = \alpha(T) \) to decrease to a new temperature.
  - If \( \text{best\_count} > 0 \) then set \( \text{frozen\_count} = 0 \)
  - If renew/iteration count \( < 1/N_{size} \) then set \( \text{frozen\_count} = \text{frozen\_count} + 1 \)
- Until outer loop stopping criterion is met

Complimentary Contributor Copy
Inner loop Procedure:
Set iteration_count = 0.
Repeat (inner loop)
   Call Neighborhood solution search procedure
to generate a feasible neighborhood solution \( x' \).
   Calculate \( f(x') \).
   Call Efficient procedure
   of calculating the cost difference \( \Delta_{x,x'} = f(x') - f(x) \).
   If \( \Delta_{x,x'} < -\delta \) then
      Set \( x = x' \), renew = renew + 1.
      Set \( f(x) = f(x') \).
      If \( f(x) < f_{\text{best}} \) then
         \( x_{\text{best}} = x \)
         \( f_{\text{best}} = f(x) \)
         best_count = best_count + 1
         Record results on “Best So Far”
      Endif
   else
      If random[0, 1] < \( \exp(-\Delta_{x,x'}/T) \) then
         \( x = x' \)
         \( f(x) = f(x') \)
         renew = renew + 1
      Endif
   Set iteration_count = iteration_count + 1.
Endif
Until stopping criteria of inner loop is met

In implementing the SA algorithm described above, initial feasible solution producing procedure, the procedure of selecting initial temperature, neighborhood solution search procedure, efficient way to calculate the difference of objective function values of two neighborhood solutions, acceptance function (here it is the Metropolis function), cooling scheduling of the temperature, and stopping criterions of inner and outer loops are its important components. Different definitions of those are discussed in the literature on SA methods, which will be discussed in the following subsections.

SA algorithm corresponds to a Markov chain. For each temperature \( T \) fixed, if the variation of Markov chain arrives at a stable state and then \( T \) goes down, we call the SA algorithm homogeneous SA algorithm; if not, it is a non-homogeneous SA algorithm.
2.2 Initial Feasible Solution Producing Procedure

For a convex function the initial feasible solution can be chosen anywhere, from which the global minimum is reached by moving towards to the lower values, in the feasible region; however, for a non-convex function, it depends on the initial solution very much either to find a local minimum of it or to find its global minimum [9]. For real projects, usually there are many requirements, i.e., constraints, for reaching its aims.

A good feasible initial solution producing procedure is clearly needed. Numerical experiments show that, without the sensitive procedure of choosing the initial simplex (see § 3.1.1 of [10]), the Simplex Simulated Annealing (SSA) method of paper [10] is very difficult to make it work. However, for many problems solved by SA method, there, often, is a simple way of producing initial solution: randomly taking a feasible solution from the feasible region as the initial solution. Constraint programming is a new high-level language paradigm for satisfaction and optimization problems. To produce a feasible solution by constraint programming strategy as the initial solution is also a very popular way; for example, see [11]. Using a local/global search optimization method to quickly get a solution as the initial solution of SA is also a good way.

2.3 Initial Temperature Selecting Procedure

Numerical experiments tell us that proper initial temperature T₀ can make the SA method quickly get the optimal value of the objective function.

If at initial temperature we accept almost all the solution (i.e., acceptance rate χ₀ ≈ 1), then in theory, by Metropolis criterion \( \exp(\Delta x/\chi_0^2) = 1 \), where T₀ should be “sufficiently” large. Johnson, Kirkpatrick, Aarts et al. [12-15, 3] present several initial temperature selecting procedures. The idea of Kirkpatrick is: to choose a large T₀, give χ₀ in advance (for example χ₀=0.8), generate many solutions, if the acceptance rate χ is less than χ₀ then increase T₀, repeat until χ > χ₀ to get a T₀. Johnson’s formula is

\[
T_0 = \frac{\Delta^- f^+}{\ln(\chi_0^{-1})},
\]

where \( \Delta^- f^+ \) is the average increase of objective function values of randomly generated solutions. The one of [12] is frequently used; for example, in [10, 8]. Aarts’ formula is
\[
T_0 = \Delta \tilde{f} / \ln \left( m_1 / ( m_2 \chi - m_1 (1-\chi) ) \right),
\]

where \(m_1\) is the number of solutions making the objective function value decrease, \(m_2\) is the number of solutions making the objective function value increase, and \(\chi\), for example, may be set as \(\chi_0\). However, those procedures are not definitely working well for all problems. Fixed temperature schedule is studied and applied in [16-18]. In homogeneous SA, \(T_0\) chosen should be properly large enough to sufficiently accept all the candidate feasible solutions possibly produced. We also often use the following ways:

1. Uniformly sample some solutions, calculate their objective function values, and take the variance of those objective function values as \(T_0\).
2. Randomly generate some solutions, determine \(|\Delta_{\text{max}}|\) which is the maximal difference of each pair of solutions, and calculate \(T_0 = -\Delta_{\text{max}} / \ln p_r\), where \(p_r\) is the initial acceptance probability and in theory it should be close to 1. For non-homogeneous SA, in theory we have formulas [19] for calculating \(T_0\).

### 2.4. Neighborhood Solution Searching Procedure

This is one key element in implementing SA. For discrete optimization problem, for instance, in the book [1], there are several successful neighborhood solution searching procedures. All those procedures should be at least based on two basic ideas: (a) neighbor means “nearby”, (b) SA method is a stochastic method so that the neighborhood solution should be randomly taken. We may take those ideas for developing neighborhood solution searching procedure for continuous optimization problems.

First we review some neighborhood solution searching procedures of continuous optimization problems. Miki et al. (1999) presented a formula \(x_{k+1}^i = x_k^i + r \cdot m\), where \(r\) is a random number with uniform distribution in \([-1,1]\), \(m\) is the neighborhood range which makes the rate between accepted and rejected moves approximate 0.5 [20]. In [21], first generate a random direction vector \(\theta_k \in \mathbb{R}^n\), with \(\|\theta_k\| = 1\), then find a fixed step size \(\Delta r\), thus get a neighborhood solution \(x_{k+1}\) of \(x_k\): \(x_{k+1} = x_k + \Delta r \cdot \theta_k\). The choice of \(\Delta r\) is thoroughly discussed in [22-23]. In [23], the direction vector \(\theta_k\) is defined in a new way. It is suggested to take into account the point \(x_h\), \(h < k\), generated by the algorithm and different from \(x_k\), if \(f(x_h) < f(x_k)\) then \(\theta_k = x_h - x_k\), otherwise \(\theta_k = x_k - x_h\). Contrary to the results in [21-22], Corana et al. (1987), Siarry et al. (1997) and Vanderbilt et
al. (1984) search through the space of feasible region in an anisotropic way [24-26]. In [24], at each iteration \( k \) a single variable of \( x_k \) is modified, and iterations are subdivided into cycles of \( n \) iterations during which each variable is modified; i.e., \( x_{vi+1} = x_v + r v_{vi+1} e_{vi+1} \), where \( r \) is a uniform random number in \([-1,1] \), \( i \in \{0, \ldots, n-1\} \) is such that \( k+1 = h*n+i \) for some nonnegative integer \( h \), and \( v_{ni+1} \) (that is anisotropic) is the maximum allowed step along the direction \( e_{vi+1} \) of the \((i+1)\)-st axis. Instead of varying a single variable in \( x_k \) in each iteration, Siarry et al. (1997) varied \( p \) variables [25]. Another concept to simulated annealing method is adaptive (see [27-32]). This means SA method should possess the ability of adapting itself to the problem it solves, the objective function \( f \) and the temperature, etc. whether globally or locally. The code of Ingber’s ASA (Adaptive Simulated Annealing) algorithm [28-31] can be retrieved from the web site www.ingber.com, and many techniques such as ‘fast annealing’, ‘re-annealing’, ‘quenching’, ‘multistart strategy’, and ‘priori information’ are used. Romeijn et al. (1994, 1999) proposed a two-phase generator: “\text{first generating a random direction } \theta_k \text{ with } ||\theta_k|| = 1, \text{ and then generating a random point } \lambda_k \text{ in the set } \Lambda_k(\theta_k) = \{\lambda: x_k + \lambda \theta_k \in X\}, \text{ thus } x_{\lambda+1} = x_k + \lambda \theta_k^\dagger \text{”, and if } x_{\lambda+1} \text{ is not in } X \text{ or if there is a jamming problem, i.e., } \Lambda_k \text{ is very small, then use the ‘reflections’ technique [33-34].} Employing computer science theory is also useful for the neighborhood solution searching procedure; for example Bilbro et al. (1991) gave a tree annealing approach [35]: divide the feasible regions in the form of a tree, and \( x_{k+1} \) is sampled from a distribution which selects a random path from the root of the tree to a leaf of the tree in a way that the sub-regions with a high percentage of accepted points are favored. Employing a local search method into simulated annealing method is also very popular. Desai et al. (1996) proposed a technique [36]: randomly perturb the current solution \( x_k \) to get a new point \( x_{k+1} \), and start a local search from \( x_{k+1} \) to get a new local minimum \( x_{k+1} \), which attempts to combine the robustness of annealing in rugged terrain with the efficiency of local optimization methods in simple search spaces. The parallel version of [36] may be seen in [37]. Lucidi et al. (1989) presented a random tunneling technique by means of acceptance-rejection sampling [38]. Over the unit hypercube feasible region, Fox (1995) gave a special neighborhood solution searching procedure [39]. First the objective function \( f \) is evaluated at the points of a net. Then, the unit hypercube is subdivided into many boxes of a set \( C \), which are with different widths along different axes. Over \( C \), generate a probability mass function \( p \) by intersecting the net with each box to generate many points and find the minimum value of those points. Then sample a box \( B \) from \( C \) according to the probability mass function \( p \), sample a uniform point \( y \)
from B and apply some local search steps starting from y. Repeat the sampling of B and y a finite number of times, and get the set F(x_k, k), which is a finite set of candidate neighborhood points of x_k at iteration k. And then the acceptance probabilities are applied to define the distribution of the next iteration L ≥ k, and according to the acceptance probabilities the algorithm randomly selects a point in F(x_k, k) and moves to it. For getting neighborhood solution, there is an idea: simultaneously perturbing all of the variables of the problem in a “proper” random fashion. In [40] the neighborhood solution producing procedure is given by the way: randomly uniformly re-generate one element of x_k, or m∈ random{1, ...,n} elements of x_k, or the whole vector of x_k, as the new solution x_{k+1}. In [10], instead of the reflect-expand-contract-shrink Nelder-Mead method (seen, for example, in [41]), the reflects-shrink Simplex Direct Search (SDS) method is given. The SDS method uses various numbers of reflections in a flexible manner and follows a shrinking if after reflecting all the n worst vertices of the initial simplex better movement is still failed to be given.

In Fast Simulated Annealing method of [42], Cauchy distribution is used to produce new solution. Greene et al. (1986) used the probability of fitness function, which is based on objective function, to produce new solution [43].

From the ideas of all those reviewed above, we present two versions of the neighborhood solution search procedure for SA algorithm. In the SDS algorithm, for the objective function value of each vertex we add a random fluctuation: f(x_i)+k_B T log(z), where k_B is the Boltzmann constant in appropriate units and z is a random number of the interval (0,1). We might carry out a multiple shrinking, in which the highest vertex is simultaneously moved along all the coordinates towards the lowest-energy vertex. For getting a new neighborhood solution, in [40], the procedure below is used:

\[
\text{\textit{i = random\{1, 2, 3\}, which is a random integer taken from the set \{1, 2, 3\}. Depending on the outcome of i, within the feasible region, re-generate randomly one of the following: one element of x, or } \text{m} \in \text{\{1, 2, \ldots, n\}} \text{ elements of x, or the whole vector of x. This gives } x'.
\]

Noticing that the neighborhood solution search for simulated annealing method should be at least based on ideas (a) and (b), we may simply give a neighborhood solution search procedure for simulated annealing algorithm:

Uniformly randomly keeping n−1 elements of x, and making the left one element of x uniformly randomly take a value such that the new solution x' is still feasible. This gives x'.
When the feasible region of the optimization problem is the unit simplex $S$, the neighborhood solution search procedure should be modified:

*Uniformly randomly keeping $n-2$ elements of $x$, and making one element from the two elements left to $x$ uniformly randomly take a value from $[0,1]$ such that the value of the sum of the $n-1$ elements is not greater than 1. Another left element of $x'$ is given the value 1-sum. This gives $x'$.*

This is an efficient procedure, which is better than the one in [40].

Another version of neighborhood solution procedure may be found in the following SA pseudo-code:

*New version of the simulated annealing algorithm.*

1. $X_{\text{best}} = x$ & $f_{\text{best}} = f$; $q := q_0$
2. DO $j := 1$ to $J$
3. $T := T_0(j)$
4. Repeat
5. DO $k := 1$ to $L$
6. Repeat
7. Randomly generate the search direction $d \in (-1, 1)$
8. Let $x'_i = x_i + q \cdot d$ and $x'_l = x_l$ when $l \neq i$
9. Until $x'$ is feasible
10. Calculate $\Delta'x$
11. IF ($\Delta'x < 0$) or ($\exp(-\Delta'x/T) > \text{random}[0, 1]$) THEN
12. Accept $x'$
13. IF $f < f_{\text{best}}$ THEN $x_{\text{best}} = x$ & $f_{\text{best}} = f$
14. Calculate next annealing temperature $T$
15. Adjust the step length $q := g(\text{Acc}) \cdot q$ (where $g(\cdot)$ is a function given)
16. END DO
17. Until outer loop stopping criterion is satisfied
18. IF $f_{\text{best}} < f$ THEN $x = x_{\text{best}}$ & $f = f_{\text{best}}$
19. END DO

where $g(\cdot)$ is an adjustment function, for example $g(x) = (x-0.5)^3 + 1$. As a whole, the new solution generating procedure composes two parts: the way to generate candidate solution, and how to generate the probability distribution of the candidate solution. Hence, we may replace our uniform distribution by Cauchy distribution, Gauss normal distribution, or their combined distribution, and get some new results for comparisons.
2.5. Efficient Calculation of Cost Difference

Since a very large number of iterations are performed, it is essential to calculate efficiently the cost differences between a solution and its neighborhood solution. Take a simple instance, if \( f(x) = Ax + b \), it is clearly \( \Delta x = A(x' - x) \) is much efficient than \( \Delta x = (Ax' + b) - (Ax + b) \), especially when the computational effort is very much for the computer.

Note: There is a subtle difference in the meaning between objective function and fitness function. Objective function measures the variable’s performance in the search region; whereas fitness function provides a measure of variable’s relative performance by transforming the objective function \( f \) into \( F(x) = g(f(x)) \). For example, proportional fitness assignment \( F(x_i) = f(x_i) / \sum f(x_i) \), linear transformation \( F(x) = af(x) + b \) are some simple transformations.

2.6. Acceptance Function

The (Markov chain state) acceptance function, generally, is given in a probability form that should meet the following criteria:

1) At each fixed temperature, the acceptance function should maintain the average percentage of accepted moves at about 1/5 of the total number of moves, which cannot make the objective function value decrease;
2) With the decrease of temperatures, the probability of accepting an increasing move decreases.
3) When temperature becomes zero, only the solutions that make the objective function value decrease can be accepted.

In Subsection 2.1, the acceptance function for the SA algorithm is the so-called Metropolis function

\[
A(x, x', T) = \min \{1, \exp \left\{ -\frac{\Delta x}{T} \right\} \}. \tag{2.1}
\]

Note here we let the acceptance function depend on the difference of the function values of \( x \) and \( x' \) instead of depending directly on \( x \) and \( x' \). More generally, we may rewrite (2.1) as follows:

\[
A(x, x', T) = \min \{1, \exp \left\{ -\frac{\Delta x}{\gamma(T)} \right\} \}, \tag{2.2}
\]
if $\gamma: (0, +\infty) \rightarrow (0, +\infty)$ is a strictly increasing function under some balance conditions [44]. Barker’s function

$$A(x, x', T) = \frac{1}{1 + \exp \{-\Delta x'/T\}} \quad (2.3)$$

is another popular acceptance function. This function has a similar form for $T$ to (2.2). Johnson et al. (1987) suggested

$$A(x, x', T) = \min\{1, 1 - \Delta x'/T\} \quad (2.4)$$

and made the speed of SA algorithm increase by 30% [45]. Sechen (1988) used table search to reduce the time wasted on calculating $\exp\{-\Delta x'/T\}$ [46].

### 2.7. Cooling Scheduling of Temperature

During the SA iterations, the temperature sequence $\{T_k\}$ is being produced. If $\lim_{k \to +\infty} T_k = 0$, we say that $\{T_k\}$ is a cooling schedule. In this subsection, we review some successful cooling schedules.

Aarts and Laarhoven (1985) presented a cooling scheme [47]

$$T_{k+1} = T_k / \left(1 + T_k \log( + \gamma) / (3\sigma_k)\right), \quad (2.5)$$

where $\sigma_k$ is the standard deviation of the observed value of the cost function, and $\gamma$ is 0.1 in [18].

$$T_{k+1} = T_k \exp\{-T_k(f_k - f_{k-1}) / \sigma^2_{T_k}\}$$

is another cooling schedule [48]. Reeves (1995) described a cooling schedule of Lundy and Mees [4], where the temperature is reduced according to

$$T_{k+1} = T_k / (1 + \beta T_k), \quad (2.6)$$

or equivalently,

$$T_k = T_0 / (1 + k\beta T_0), \quad (2.7)$$

where $\beta$ is suitably small, and only one iteration is performed in each inner loop. For the convergence of non-homogeneous SA method, in 1984 Geman et
In Mathematical Global Optimization Computation …

al. (1984) gave the Boltzmann annealing or called classical SA [49], in which the temperature is calculated by

\[ T_k = T_0 / \ln(k + c), \quad k = 1, \ldots, \infty, \quad (2.8) \]

where \( c = 1 \). A little modification of \( c \) is used in [50]:

\[ T_k = T_0 / \ln(k + c), \quad k = 0, \ldots, \infty \]

(2.9)

with \( c = e = 2.7183 \). For formulas (2.8) and (2.9) \( c \) should not be less than 1. In 1987, Szu et al. (1987) proposed the Fast Annealing method [42]. The cooling schedule of this method is with a faster decrease:

\[ T_k = T_0 / (k + 1), \quad k = 1, \ldots, \infty \]

(2.10)

that decreases sharper than (2.8). However, we should match the rate of temperature decrease with the neighborhood solution generating procedure. Nahar et al. (1987) divided \([0, T_0]\) into \( K \) intervals and find \( T_k \), \( k = 1, \ldots, K \) [15]. The Very Fast Simulated Re-annealing method [28] was presented in 1989 by Ingber. Its cooling schedule is

\[ T_k = T_0 \exp(-ck^{1/n}), \quad k = 1, \ldots, \infty, \]

(2.11)

where \( c \) is a scale factor. Ingber (1989) also used a slower schedule [28] of (2.10), which is

\[ T_k = T_0 / (k + 1)^{1/n}, \quad k = 1, \ldots, \infty. \]

(2.12)

Although many cooling schedules are mentioned above, the geometric cooling scheme [3] proposed by Kirkpatrick et al. (1983)

\[ T_{k+1} = \alpha T_k, \quad k = 0, \ldots, \infty, \]

(2.13)

where \( \alpha \in (0, 1) \) is a constant, is still a widely used and a popular SA cooling schedule (refer to [12-15, 10, 4, 40] because it compromises the quality and CPU time of optimization. Kirkpatrick et al. (1983) take \( \alpha = 0.95 \); and Johnson et al., take \( \alpha \in [0.5, 0.99] \). Our numerical experiments also shows that \( (0.8, 0.99) \) is a good interval chosen for \( \alpha \). Given \( T_0 \), \( T_f \) and the number of outer loop iterations, in [51] graphs of many kind of cooling schedules can be seen.
2.8. Stopping Criterion of Inner Loops

The number of iterations in each inner loop is also called the temperature length. In many forms of simulated annealing method, a fixed number of iterations are performed for each temperature. Usually the fixed number is detected by a long sequence of iterations in which no new solutions have been accepted. This fixed number depends on the size of the neighborhood Nsize, which is defined to be the total number of possible distinct neighborhood solutions that can be obtained, and its mathematical form is Nfactor*Nsize, where Nfactor is some multiplying factor, for example Nfactor=10. In our pseudo-code of the simulated annealing method, we also introduce a symbol renew which records the number of times the solutions are accepted at a temperature. We may also terminate the inner loop if this number has exceeded Cut*Nfactor*Nsize where Cut is another multiplying factor. Section 4.2.3 of [9] describes this fixed number in view of stochastic process terminologies. In homogeneous SA, from the view of objective function values two stopping criteria may be presented:

1) Checking stability of the expectation value of objective function values; and
2) The change of objective function values is lower than some tolerance in a certain amount of iterations.

2.9. Stopping Criterion of Outer Loops

The choice of final temperature $T_f$ determines a stopping criterion of outer loops. At the end of each inner loop, if the best solution obtained in that inner loop has not been changed and at the same time we are not having many changes in the current solution, we reduce the temperature and start another inner loop. If a solution $x^*$ has been consecutively “frozen” at many current temperatures, then we stop and say $x^*$ is the best solution found by the simulated annealing method. Usually we set a proper small temperature as $T_f$ as the stopping condition of outer loops. Our numerical experiments show that we may get the small temperature $T$ when “Floating point exception (core dumped)” is reminded by computer. In our pseudo-code, we also count the number $\text{best\_count}$ of times at which the best feasible solution is not replaced again, and calculate the proportion of solutions accepted, by $\text{frozen\_count}$. We may halt the algorithm when $\text{frozen\_count}$ reaches a predetermined number.
As a whole, based on \( \lim_{k \to +\infty} T_k = 0 \), we find a “frozen” temperature for the stopping criterion of outer loops. Nahar et al. (1987) used the number of temperatures, in other words, the number of Markov chains or iterations, as stopping criterion of outer loops [15]. Notice here the lengths of Markov chains, \( L_k \), may be upper bounded by a constant \( \bar{l} \). Johnson et al. (1987) used the acceptance rate to terminate the outer loops [45]: current acceptance rate \( \chi_k > \chi_f \) given, where \( \chi_f \) is the final acceptance rate.

From the point of view of the objective function values, we may also give the terminating criterion for the outer loops. If \( |\Delta x| \leq \varepsilon \) or \( f(x) - f_{\text{best}} \leq \varepsilon \), \( (f_{\text{best}} - f_{\text{opt}})/f_{\text{opt}} \leq \varepsilon \) (where \( |f_{\text{best}} - f_{\text{opt}}| < \varepsilon \) if \( f_{\text{opt}} = 0 \)), where \( \varepsilon \) is a sufficiently small positive number and \( f_{\text{opt}} \) is the optimal value known, we stop the method. This simply means when the objective function values cannot be improved we may stop the algorithm.

In another form, we use the information not only on the temperature but also on the objective function values; then we can also give a stopping criterion for the simulated annealing method. Suppose \( P_F \) is a proper number given, if \( A(x, x', T) \leq P_F \), we stop the simulated annealing method. If in many successive Markov chains the solution has not changed, we can stop the method.

Setting an upper limit of executing time is also a way to stop the algorithm. The user may terminate the method manually according to a user-defined aim.

### 2.10. Improvements on SA Method

Aarts et al. (1989) improved the simple cooling scheme of Johnson et al., talked in the above Subsection 2.7 (see [5]). They present a more meticulous cooling scheme in which \( T_0, T_f, L_k \) and the formula of \( T_k \) are well designed. Other improvements on SA are:

1. Re-increase temperature. In the process of SA, in order to adjust some state, to reincrease its temperature is a good way.

   **Heating-annealing procedure.**
   
   \[
   T_0 = 0 \\
   \text{Repeat} \\
   \text{DO } k := 1 \text{ to } L \\
   \text{Generate new solution } x' 
   \]
Calculate $\Delta_{x^}\!$x
IF $\Delta_{x^}\!x > 0$ THEN accept $x^\prime$ and $HEAT$: $T := \text{heat}(T)$;
IF ($\Delta_{x^}\!x < 0$) or (exp($-\Delta_{x^}\!x/T$) > random[0, 1]) THEN accept $x^\prime$
END DO
IF $HEAT$ THEN exit, ELSE calculate next annealing temperature $T$
Until outer loops stopping criterion is satisfied where $L$ is the length of Markov chain at $T$.

(2). Catch messages on “Best So Far”, inserting into inner and outer loops a procedure:

If improve “Best So Far”, then index:=0;
Otherwise, index:=index+1.

More in details, the procedure is:

$Memory$-$Annealing$ procedure.
x_best = x & f_best = f
Repeat
DO k := 1 to L
Generate new solution $x^\prime$
Calculate $\Delta_{x^}\!x$
IF ($\Delta_{x^}\!x < 0$) or (exp($-\Delta_{x^}\!x/T$) > random[0, 1]) THEN
Accept $x^\prime$
IF $f < f_{\text{best}}$ THEN $x_{\text{best}} = x$ & $f_{\text{best}} = f$
END IF
END DO
Calculate next annealing temperature $T$
Until outer loops stopping criterion is satisfied
$x = x_{\text{best}} & f = f_{\text{best}}$

(3). At the end of SA, take the optimal solution obtained as the initial solution to carry on a local search method or the SA method again.

$Annealing$-$Local$ $Search$ procedure.
x_best = x & f_best = f; anneal=true
Repeat
Repeat
DO k := 1 to L
Generate new solution $x^\prime$ (Randomly local search or Allover local search)
Calculate $\Delta_{x^}\!x$
IF ($\Delta_{x^}\!x < 0$) or (anneal & exp($-\Delta_{x^}\!x/T$) > random[0, 1])) THEN
Accept $x^\prime$
IF f < f_best THEN x_best = x & f_best = f
END IF
END DO
IF anneal THEN calculate next annealing temperature T
Until outer loops stopping criterion is satisfied
IF f_best < f THEN x = x_best & f = f_best
anneal := not (anneal)
Until anneal

In [52] after Accept x’ a downhill local search method is embedded. On the contrary, before SA search we also may carry on a local search:

Local Search-Annealing procedure.
x_best = x & f_best = f; search = true & m := 0
Repeat
IF search THEN L := Ls & m := m + 1 ELSE L := Lh
Repeat
a := 0
DO k := 1 to L
Generate new solution x’ (Random local search or Allover local search)
Calculate Δ_x
IF (search & Δ_x < 0) or ( NOT search & Δ_x > 0 & exp(−Δ_x/T) > random [0, 1] ) THEN
Accept x’
a := 1
END IF
END DO
Until (search & (a = 0)) or (NOT search & (a = 1))
IF f_best < f THEN x = x_best & f = f_best
anneal := not (anneal)
search := NOT search
Until m = snum
x = x_best & f = f_best
where snum is the number of optimal searches given.

(4). During the SA, for current state, take several search strategies, and accept the best state found with respect to probability.
3. THE HYBRID OF SA WITH A LOCAL/GLOBAL SEARCH METHOD

Global optimization SA search sometimes trapped at local minima and cannot reach the real global minima. Hybrid with local search or global search optimization method is a strategy to bring the SA out of the trapped local minima. In this Section we introduce several successfully tested hybrid methods of SA. In Subsection 3.1, we will introduce the hybrid SA with local search discrete gradient (DG) method, and in Subsection 3.2 we will introduce the hybrids of SA with global search Self-Adaptive Evolutionary Strategy μ+λ (SAES(μ+λ)) method and global search Self-Adaptive Classical Evolutionary Programming (SACEP) method.

3.1. Hybrid SA Discrete Gradient Method

3.1.1. Efficiency of Discrete Gradient Method

The DG method [53] is a derivative-free local search optimization method. Therefore, first we investigate the efficiency of discrete gradient method, comparing with other well-known derivative-free methods. We use small-size standard test problems to test the DG method, Nelder-Mead’s simplex method [54], and Powell’s UOBYQA method [54]. For each problem and each dimension, we run the three methods 50 times. The 50 initial solutions are randomly taken from the feasible region. The best optimal value obtained and its frequency of occurrence, the mean and the variance of 50 optimal values obtained can be seen in the following database.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Dimension</th>
<th>Method</th>
<th>Best value obtained</th>
<th>Frequency</th>
<th>Mean</th>
<th>Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Camel [55]</td>
<td>2</td>
<td>DG</td>
<td>-1.031628</td>
<td>80%</td>
<td>-0.86840</td>
<td>0.10876</td>
</tr>
<tr>
<td>Simplex</td>
<td></td>
<td></td>
<td>-1.031628</td>
<td>82%</td>
<td>-0.85462</td>
<td>0.26388</td>
</tr>
<tr>
<td>UOBYQA</td>
<td></td>
<td></td>
<td>-1.031628</td>
<td>46%</td>
<td>-0.17335</td>
<td>1.29922</td>
</tr>
<tr>
<td>Goldstein-Price [56]</td>
<td>2</td>
<td>DG</td>
<td>3.000000</td>
<td>50%</td>
<td>65.10002</td>
<td>2.6E+04</td>
</tr>
<tr>
<td>Simplex</td>
<td></td>
<td></td>
<td>3.000000</td>
<td>60%</td>
<td>372.30849</td>
<td>2.9E+06</td>
</tr>
<tr>
<td>UOBYQA</td>
<td></td>
<td></td>
<td>3.000000</td>
<td>40%</td>
<td>122.88000</td>
<td>7.2E+04</td>
</tr>
<tr>
<td>Griewank [41]</td>
<td>6</td>
<td>DG</td>
<td>9.224853</td>
<td>2%</td>
<td>26.62373</td>
<td>121.67116</td>
</tr>
<tr>
<td>Simplex</td>
<td></td>
<td></td>
<td>0.277595</td>
<td>2%</td>
<td>7.6E+04</td>
<td>2.9E+11</td>
</tr>
<tr>
<td>UOBYQA</td>
<td></td>
<td></td>
<td>0.946213</td>
<td>2%</td>
<td>81.79783</td>
<td>1764.17313</td>
</tr>
<tr>
<td>Hansen [57]</td>
<td>2</td>
<td>DG</td>
<td>-176.541793</td>
<td>94%</td>
<td>-174.67794</td>
<td>55.53534</td>
</tr>
<tr>
<td>Simplex</td>
<td></td>
<td></td>
<td>-176.541793</td>
<td>44%</td>
<td>-134.99624</td>
<td>2652.23300</td>
</tr>
</tbody>
</table>

Numerical results for the DG method, Simplex method and UOBYQA method

Complimentary Contributor Copy
<table>
<thead>
<tr>
<th>Function</th>
<th>Method</th>
<th>Dim</th>
<th>Best Value</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hartman</td>
<td>UOBYQA</td>
<td>3</td>
<td>-176.541793</td>
<td>30%</td>
</tr>
<tr>
<td></td>
<td>Simplex</td>
<td></td>
<td>-3.86278</td>
<td>0.00000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>76%</td>
<td>0.33637</td>
</tr>
<tr>
<td></td>
<td>UOBYQA</td>
<td>3</td>
<td>-3.86278</td>
<td>80%</td>
</tr>
<tr>
<td></td>
<td>Simplex</td>
<td></td>
<td>-3.71331</td>
<td>0.00056</td>
</tr>
<tr>
<td></td>
<td>UOBYQA</td>
<td>6</td>
<td>-3.322368</td>
<td>96%</td>
</tr>
<tr>
<td></td>
<td>Simplex</td>
<td></td>
<td>-3.31760</td>
<td>0.00056</td>
</tr>
<tr>
<td></td>
<td>UOBYQA</td>
<td>5</td>
<td>-3.322337</td>
<td>8%</td>
</tr>
<tr>
<td></td>
<td>Simplex</td>
<td></td>
<td>-2.81436</td>
<td>0.32494</td>
</tr>
<tr>
<td></td>
<td>UOBYQA</td>
<td>3</td>
<td>-3.322368</td>
<td>84%</td>
</tr>
<tr>
<td></td>
<td>Simplex</td>
<td></td>
<td>-3.30329</td>
<td>0.00195</td>
</tr>
<tr>
<td></td>
<td>Levy Nr.1</td>
<td>2</td>
<td>0.0000000</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>Simplex</td>
<td></td>
<td>1.80857</td>
<td>9.23844</td>
</tr>
<tr>
<td></td>
<td>UOBYQA</td>
<td>10</td>
<td>4.19420</td>
<td>96%</td>
</tr>
<tr>
<td></td>
<td>Simplex</td>
<td></td>
<td>1.55279</td>
<td>2.71983</td>
</tr>
<tr>
<td></td>
<td>UOBYQA</td>
<td>20</td>
<td>0.0000000</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>Simplex</td>
<td></td>
<td>1.24647</td>
<td>0.48673</td>
</tr>
<tr>
<td></td>
<td>Levy Nr.2</td>
<td>2</td>
<td>0.0000000</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>Simplex</td>
<td></td>
<td>1.6E+08</td>
<td>3.0E+16</td>
</tr>
<tr>
<td></td>
<td>UOBYQA</td>
<td>10</td>
<td>4.277692</td>
<td>86%</td>
</tr>
<tr>
<td></td>
<td>Simplex</td>
<td></td>
<td>2.1E+08</td>
<td>5.40594</td>
</tr>
<tr>
<td></td>
<td>UOBYQA</td>
<td>20</td>
<td>0.0000000</td>
<td>12%</td>
</tr>
<tr>
<td></td>
<td>Simplex</td>
<td></td>
<td>4.277692</td>
<td>2%</td>
</tr>
<tr>
<td></td>
<td>Levy Nr.3</td>
<td>4</td>
<td>-21.502355</td>
<td>28%</td>
</tr>
<tr>
<td></td>
<td>Simplex</td>
<td></td>
<td>-10.64044</td>
<td>108.3934</td>
</tr>
<tr>
<td></td>
<td>UOBYQA</td>
<td>5</td>
<td>-18.392864</td>
<td>34%</td>
</tr>
<tr>
<td></td>
<td>Simplex</td>
<td></td>
<td>41.83276</td>
<td>1751.9511</td>
</tr>
<tr>
<td></td>
<td>UOBYQA</td>
<td>5</td>
<td>-11.504402</td>
<td>14%</td>
</tr>
<tr>
<td></td>
<td>Simplex</td>
<td></td>
<td>-10.98174</td>
<td>0.25725</td>
</tr>
<tr>
<td></td>
<td>UOBYQA</td>
<td>20</td>
<td>-8.505954</td>
<td>52%</td>
</tr>
<tr>
<td></td>
<td>Simplex</td>
<td></td>
<td>14.02895</td>
<td>210.25049</td>
</tr>
<tr>
<td></td>
<td>Levy Nr.4</td>
<td>4</td>
<td>-10.153199</td>
<td>2%</td>
</tr>
<tr>
<td></td>
<td>Simplex</td>
<td></td>
<td>-9.85005</td>
<td>1.46911</td>
</tr>
<tr>
<td></td>
<td>UOBYQA</td>
<td>10</td>
<td>-10.153200</td>
<td>16%</td>
</tr>
<tr>
<td></td>
<td>Simplex</td>
<td></td>
<td>-6.11971</td>
<td>12.29406</td>
</tr>
<tr>
<td></td>
<td>UOBYQA</td>
<td>10</td>
<td>-10.153200</td>
<td>74%</td>
</tr>
<tr>
<td></td>
<td>Simplex</td>
<td></td>
<td>-8.44760</td>
<td>8.82110</td>
</tr>
<tr>
<td></td>
<td>UOBYQA</td>
<td>4</td>
<td>-10.402940</td>
<td>96%</td>
</tr>
<tr>
<td></td>
<td>Simplex</td>
<td></td>
<td>-10.14444</td>
<td>1.69414</td>
</tr>
<tr>
<td></td>
<td>UOBYQA</td>
<td>10</td>
<td>-10.402937</td>
<td>10%</td>
</tr>
<tr>
<td></td>
<td>Simplex</td>
<td></td>
<td>-4.11194</td>
<td>8.81334</td>
</tr>
<tr>
<td></td>
<td>UOBYQA</td>
<td>10</td>
<td>-10.402940</td>
<td>26%</td>
</tr>
<tr>
<td></td>
<td>Simplex</td>
<td></td>
<td>-5.29144</td>
<td>9.35501</td>
</tr>
<tr>
<td></td>
<td>UOBYQA</td>
<td>20</td>
<td>-10.536410</td>
<td>96%</td>
</tr>
<tr>
<td></td>
<td>Simplex</td>
<td></td>
<td>-10.26701</td>
<td>1.85441</td>
</tr>
<tr>
<td></td>
<td>UOBYQA</td>
<td>10</td>
<td>-10.536410</td>
<td>20%</td>
</tr>
<tr>
<td></td>
<td>Shubert Nr.1</td>
<td>2</td>
<td>-186.730909</td>
<td>92%</td>
</tr>
<tr>
<td></td>
<td>Simplex</td>
<td></td>
<td>-179.82264</td>
<td>691.82815</td>
</tr>
<tr>
<td></td>
<td>UOBYQA</td>
<td>20</td>
<td>-186.730909</td>
<td>26%</td>
</tr>
<tr>
<td></td>
<td>Simplex</td>
<td></td>
<td>-78.58946</td>
<td>5030.09156</td>
</tr>
<tr>
<td></td>
<td>UOBYQA</td>
<td>20</td>
<td>-186.730909</td>
<td>18%</td>
</tr>
<tr>
<td></td>
<td>Simplex</td>
<td></td>
<td>-168.91354</td>
<td>235.63292</td>
</tr>
<tr>
<td></td>
<td>UOBYQA</td>
<td>20</td>
<td>-186.730909</td>
<td>9%</td>
</tr>
<tr>
<td></td>
<td>Simplex</td>
<td></td>
<td>-115.19069</td>
<td>3686.41945</td>
</tr>
<tr>
<td></td>
<td>UOBYQA</td>
<td>20</td>
<td>-186.730909</td>
<td>4%</td>
</tr>
<tr>
<td></td>
<td>Simplex</td>
<td></td>
<td>-28.78545</td>
<td>5083.27183</td>
</tr>
<tr>
<td></td>
<td>UOBYQA</td>
<td>20</td>
<td>-24.062499</td>
<td>42%</td>
</tr>
<tr>
<td></td>
<td>Simplex</td>
<td></td>
<td>-22.59130</td>
<td>1.59932</td>
</tr>
<tr>
<td></td>
<td>UOBYQA</td>
<td>20</td>
<td>-24.062499</td>
<td>12%</td>
</tr>
</tbody>
</table>

By the comparative analysis from the above database, we find that the discrete gradient method is the best one among all those methods not only for low dimension problems but also for higher dimension problems. Nelder-Mead’s simplex method cannot work well for the problems with dimensions.
greater than 10 and Powell’s UOBYQA method cannot work fast for the problems with dimensions greater than 20. So, we choose the DG method.

The performance profile is a cumulative distribution function over a performance ratio and provides condensed information in terms of robustness, efficiency and quality of solution information. We briefly write the two formulae as follows:

\[
\rho_{p,s} = \begin{cases} 
    t_{p,s} / \min \{ t_{p,s} : 1 \leq s \leq n_s \} & \text{if } \left| (o_{p,s} - b_p) / b_p \right| \leq \delta, \\
    \rho_{p,s} = \rho_M, & \text{otherwise}
\end{cases}
\]

where \( P \) is a given set of problems \( p=1, \ldots, n_p \), \( s \) is one of solvers \( s=1, \ldots, n_s \), \( t_{p,s} \) is the solver resource (e.g. computational time) spent by solver \( s \) on problem \( p \), \( o_{p,s} \) denotes the solution value found by solver \( s \) for problem \( p \), \( b_p \) is the best solution value found when applying all solvers to problem \( p \), \( \delta > 0 \) is a user-defined relative objective value difference threshold, \( \rho_M \) is an upper bound on \( \rho_{p,s} \) over all problem & solver pairs \( p, s \) in which solver \( s \) fails to solve problem \( p \), \( 1 \leq \tau \leq \infty \), and \( C \{ \cdot \} \) denotes the cardinality (size) of the set \{ \cdot \}. The function \( \rho_{p,s} \) is called a performance ratio and the function \( p_s(\tau) \) is called the performance profile function of the performance ratio. Our numerical experiments for the formula of \( \rho_{p,s} \) are: (i) when \( b_p \) is zero, in the denominator we replace \( b_p \) by 1, and (ii) \( \delta \leq 10^{-4} \). The performance profile of the DG method, Simplex method and UOBYQA method showed that, being compared with the Simplex method and UOBYQA method, the DG method is absolutely the winner for solver resource and always better than the UOBYQA method for all the solver resource. This gave an explanation why we had chosen the DG method to use in this Subsection.

3.1.2. Hybrid DG-SA-DG algorithm

In this subsection we develop a hybrid SA and DG for solving the global optimization problem

Minimize \( f(x) \) subject to \( x \in X \),

where \( X \) is a subset of \( \mathbb{R}^n \), \( X \) is a compact set and \( f \) is continuous and it is also a locally Lipschitz continuous function. The hybrid method starts from an initial point, first executes the DG method to find local minimum, then carries on with the SA method in order to escape from this local minimum and to find a new starting point for DG method. Then we again apply the discrete gradient method starting from the current best point and so on until the sequence of the

Complimentary Contributor Copy
optimal objective function values obtained is convergent. The pseudo-code of the hybrid method is listed as following:

\textbf{Algorithm: Hybrid discrete gradient and simulated annealing method.}

\textbf{Initialization:}
Define the objective function $f$ and its feasible solution space.
Call initial feasible solution generating procedure to get $x$.
Call initial temperature selecting procedure to get $T_0$.
Initialize $f$: $f = f(x)$.
Initialize the neighborhood feasible solution $x_{\text{neighbour}} = 0$.
Initialization of $x_{\text{best}}$: $x_{\text{best}} = x$.
Initialization of $f_{\text{best}}$: $f_{\text{best}} = f$.
\begin{verbatim}
do {
\begin{verbatim}
DG local search part:
F_{\text{best\_local}} = \text{local\_search}(x_{\text{best}}, x_{\text{new}});
x = x_{\text{new}};
\end{verbatim}
\end{verbatim}
SA global search part:
do {
\begin{verbatim}
x_{\text{neighbour}} = \text{randomly\_perturb}(x);
f_{\text{neighbour}} = f(x_{\text{neighbour}});
Calculate the difference $\Delta = f_{\text{neighbour}} - f$;
If ($\Delta \leq 0$) or (random[0,1] < exp(-$\Delta$/Temperature))
x = x_{\text{neighbour}} f = f_{\text{neighbour}};
If ($f \leq f_{\text{best}}$) x_{\text{best}} = x f_{\text{best}} = f;
\end{verbatim}
\end{verbatim}
} while (equilibrium has not been reached);
Temperature annealing
\end{verbatim}
} while (Simulated Annealing stop criterion has not been met);
\end{verbatim}
} while ($f_{\text{best}} - f_{\text{best\_local}} \leq -0.001$);

The convergence of the proposed hybrid method directly follows from the convergence of SA method and DG method. Generally, local search DG method makes the objective function value decrease a little bit from the initial guess, then the global search SA makes the value a big decreasing, the iterations go on until both the local and global searches cannot change the objective function value very much.

3.1.3. \textbf{Implementations}

The Description of problems
To examine the performance of Algorithm 14, we apply it to solve the wellknown complicated problems of Ackley, Bohachevsky, Branin, De Joung,
Easom, Goldstein and Price, Griewank, Hartman, Hump, Hyper-Ellipsoid, Levy Nr.1, Levy Nr.2, Michalewicz, Neumaier Nr.2, Neumaier Nr.3, Rastringins, Rosenbrock, Schaffer Nr.1, Schaffer Nr.2, Shekel-N, Shubert Nr.1, Shubert Nr.2, Sphere, Step, Zakharov, Zimmermanns, which can be found from [40-41, 56, 58-59].

**Implementation of algorithm**

For the SA part of this hybrid method, we use the Neighborhood solution search procedure described at the end of Subsection 2.4. We still use $T = 0.9^t T$ as the cooling schedule in this hybrid method. The initial temperature is taken large enough according to the rule in [3]. The number of inner and outer iterations are taken to be large enough guaranteeing sufficient iterations. The DG method used here reduces the constrained minimization problem to unconstrained using exact penalty functions, and it terminates when the distance between the approximation to the subdifferential and origin is less than a given tolerance $\varepsilon > 0$ ($\varepsilon=10^{-4}$). The initial solution for the hybrid method is randomly taken from the feasible region of the problem.

**Results of numerical experiments and discussions**

Numerical experiments have been carried out in VPAC (Victorian Partnership for Advanced Computing) with CPU 833MHz. The results of numerical experiments are listed as follows. We denote HDGSAM the hybrid DG and SA method.

**Numerical results for HDGSAM**

<table>
<thead>
<tr>
<th>Function obtained</th>
<th>Known evaluations</th>
<th>Number of function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ackleys 2 0.00008 0.200064</td>
<td>3.0.00012 0.200048</td>
<td>5.0.00009 0.200056</td>
</tr>
<tr>
<td>7.0.00008 0.200039</td>
<td>10.0.00003 0.200043</td>
<td>20.0.00052 0.2000623</td>
</tr>
<tr>
<td>30.0.00018 0.7000725</td>
<td>Bohachevsky 1 2 0.000000 0.500020</td>
<td>Bohachevsky 2 2 0.000000 0.500018</td>
</tr>
<tr>
<td>Bohachevsky 3 2 0.000000 0.500017</td>
<td>Branin 2 0.397887 0.397887 3.00018</td>
<td>De Joung 3 0.000000 0.500017</td>
</tr>
<tr>
<td>Easom 2-1.000000 1.20890</td>
<td>Goldstein Price 2 3.0000000 3.200016</td>
<td>Griewank 1 0.000000 0.2068</td>
</tr>
<tr>
<td>2.0.000000 0.2071</td>
<td>3.0.000000 0.2071</td>
<td>4.0.000000 0.2071</td>
</tr>
<tr>
<td>Function</td>
<td>Dimension</td>
<td>10</td>
</tr>
<tr>
<td>------------</td>
<td>-----------</td>
<td>----</td>
</tr>
<tr>
<td>Hartmann 3</td>
<td>-3.86278215</td>
<td>-3.86278215</td>
</tr>
<tr>
<td>Hump 2</td>
<td>0.0000000</td>
<td>0</td>
</tr>
<tr>
<td>Hyper-Ellipsoid</td>
<td>30</td>
<td>0.000000</td>
</tr>
<tr>
<td>Levy 1</td>
<td>5</td>
<td>0.0000000</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.0000000</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>0.0000000</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>0.0000000</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.0000000</td>
</tr>
<tr>
<td></td>
<td>70</td>
<td>0.0000000</td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>0.0000000</td>
</tr>
<tr>
<td></td>
<td>90</td>
<td>0.0000000</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>0.0000000</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>0.0000000</td>
</tr>
<tr>
<td></td>
<td>300</td>
<td>0.0000000</td>
</tr>
<tr>
<td></td>
<td>400</td>
<td>0.0000000</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>0.0000000</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>0.0000000</td>
</tr>
<tr>
<td></td>
<td>2000</td>
<td>0.0000000</td>
</tr>
<tr>
<td>Levy 2</td>
<td>5</td>
<td>0.0000000</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.0000000</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>0.0000000</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>0.0000000</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.0000000</td>
</tr>
<tr>
<td></td>
<td>70</td>
<td>0.0000000</td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>0.0000000</td>
</tr>
<tr>
<td></td>
<td>90</td>
<td>0.0000000</td>
</tr>
<tr>
<td></td>
<td>100</td>
<td>0.0000000</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>0.0000000</td>
</tr>
<tr>
<td></td>
<td>300</td>
<td>0.0000000</td>
</tr>
<tr>
<td></td>
<td>400</td>
<td>0.0000000</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>0.0000000</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>0.0000000</td>
</tr>
<tr>
<td></td>
<td>2000</td>
<td>0.0000000</td>
</tr>
<tr>
<td>Michalewicz 2</td>
<td>-1.8013</td>
<td>-1.8013</td>
</tr>
<tr>
<td>Neumaier 2</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>Neumaier 3</td>
<td>10</td>
<td>-210</td>
</tr>
<tr>
<td>Rastrigin 2</td>
<td>0.0000000</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.0000000</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0.0000000</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>0.0000000</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.0000000</td>
</tr>
<tr>
<td>Rosenbrock 2</td>
<td>0.0000000</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0.0000000</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.0000000</td>
</tr>
<tr>
<td>Schaffer 1</td>
<td>2</td>
<td>0.000026</td>
</tr>
<tr>
<td>Schaffer 2</td>
<td>2</td>
<td>0.0000000</td>
</tr>
<tr>
<td>Schelk (N=5)</td>
<td>-10.1530</td>
<td>-10.15320</td>
</tr>
<tr>
<td>Schelk (N=7)</td>
<td>-10.40294</td>
<td>-10.40294</td>
</tr>
<tr>
<td>Schelk (N=10)</td>
<td>-10.53641</td>
<td>-10.53641</td>
</tr>
<tr>
<td>Shubert 1</td>
<td>-186.7309088</td>
<td>-186.7309088</td>
</tr>
</tbody>
</table>
We can see the best objective function value obtained and the best objective function value known are equal to each other for every problem. This means our hybrid method is good and accurate for all those well-known optimization problems. Regarding the computational CPU time of the hybrid method for solving all those problems, it is very satisfactory. Take the Levy Nr.2 function as an example, the optimization method with 100 variables and 100 constraints needs 33.42019200 seconds to reach its optimal value 0.00000001, with 1000 variables and 1000 constraints needs 584.27654400 seconds to reach its optimal value 0.00000041, and even for the optimization problem with 10000 variables and 10000 constraints it needs 13696.099664 seconds to reach an objective function value 0.52427744. The runtime is on exponential increase. Based on results of numerical experiments, we can conclude for the hybrid DG and SA method: the hybrid DG and SA method is effective for many well-known optimization problems.

3.2. Hybrid SA Evolutionary Algorithms

In this Subsection, we present two hybrid SA and evolutionary algorithms. Numerical results show that all these hybrid methods of evolutionary computation algorithms work well. The numerical results show us SAES(μ+λ) method [60] and SACEP method [60] can successfully work for all our test problems. The SA algorithm is a sequential computing algorithm and evolutionary algorithms are parallel computing algorithms. So, in this Subsection, using SA method, we improve them. We use SA as a search operator once for SAES(μ + λ) method, and once for SACEP method. Both the algorithms designed in this section simply work by applying the SA on all individuals in the population of the initial generation. In subsequent generations, SA is applied only for the best solutions found so far.
Algorithm. SA-SAES($\mu + \lambda$).
Step 0. Randomly generate $\mu$ parents, where each parent $z_{k}=(x_{k}, \sigma_{k})$.
Step 1. Apply SA on each parent $x_{k}$.
Step 2. Set $\tau=\sqrt{2\sqrt{n}}$ and $\tau'=(\sqrt{2n})^{-1}$.
Step 3. Until $\lambda$ children are generated, do
Step 4. Select two parents $z_{k}=(x_{k}, \sigma_{k})$ and $z_{l}=(x_{l}, \sigma_{l})$ at random to generate child $y_{j}=(x_{j}, \sigma_{j})$.
Step 5. Discrete recombination: for each variable $x_{ji}$ and step size $\sigma_{ji}$ in $y_{j}$, do
$$x'_{ji}=x_{ji}+\sigma_{ji}N_{j}(0, 1)$$
$$\sigma'_{ji}=\sigma_{ji}\exp(\tau'N(0, 1)+\tau N_{j}(0, 1))$$
Step 6. Mutation: For each $x_{ji}$ and step size $\sigma_{ji}$ in $y_{j}$
$$x'_{ji}=x_{ji}+\sigma_{ji}N_{j}(0, 1)$$
$$\sigma'_{ji}=\sigma_{ji}\exp(\tau'N(0, 1)+\tau N_{j}(0, 1))$$
Step 7. If the number of children is less than $\lambda$, go to Step 4.
Step 8. Select the best $\mu$ individuals among all the $\mu + \lambda$ parents and children.
Step 9. Apply SA on the best individual among the selected $\mu$ individuals.
Step 10. If the stopping criteria are satisfied, stop, else go to step 2.

Algorithm. SA-SACEP.
Step 0. Randomly generate $\mu$ parents and evaluate them, where each parent $z_{k}=(x_{k}, \sigma_{k})$.
Step 1. Apply SA on each parent $x_{k}$.
Step 2. Set $\tau=\sqrt{2\sqrt{n}}$ and $\tau'=(\sqrt{2n})^{-1}$.
Step 3. For each parent, generate a child as follows
$$x'_{ji}=x_{ji}+\sigma_{ji}N_{j}(0, 1)$$
$$\sigma'_{ji}=\sigma_{ji}\exp(\tau'N(0, 1)+\tau N_{j}(0, 1))$$
Step 4. Evaluate all children
Step 5. Undertake a tournament $y$ for each parent and child as follows: select $\zeta$ individuals with replacement from the joint set of parents and children. For each individual $z$ of the $\zeta$ individuals, if $y$ is better than $z$, add 1 to the fitness of $y$.
Step 6. Select the best $\mu$ individuals among all parents and children with the highest fitness.
Step 7. Apply SA on the best individual among the selected $\mu$ individuals.
Step 8. If the stopping criteria are satisfied, stop, else go to step 1.

Numerical results listed as follows show that, from a point of view of the optimal objective function values obtained, the algorithms presented in this section separately improve SAES($\mu + \lambda$) method and SACEP method greatly.
The Optimal objective function values of SAES($\mu + \lambda$) Algorithm and SA-SAES($\mu + \lambda$) Algorithm, and SACEP Algorithm and SA-SACEP Algorithm

<table>
<thead>
<tr>
<th>Function Number</th>
<th>Number of variables</th>
<th>SAES($\mu + \lambda$)</th>
<th>SA-SAES($\mu + \lambda$)</th>
<th>SACEP</th>
<th>SA-SACEP</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1 [61]</td>
<td>2</td>
<td>-186.731</td>
<td>-186.731</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F2 [62]</td>
<td>5</td>
<td>1.0 1.0 1.0 1.0 1.0</td>
<td>1.02574 1.01 1.3336 1.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F3</td>
<td>2</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F4</td>
<td>5</td>
<td>2.4 2.4 2.4 2.4 2.4</td>
<td>2.4 2.4 2.4 2.4 2.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F5</td>
<td>3</td>
<td>1.9 1.9 1.9 1.9 1.9</td>
<td>1.9 1.9 1.9 1.9 1.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F6</td>
<td>2</td>
<td>0.2 0.2 0.2</td>
<td>0.2 0.2 0.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F7</td>
<td>2</td>
<td>0.3 0.3 0.3</td>
<td>0.3 0.3 0.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F8</td>
<td>2</td>
<td>0.1 0.1 0.1</td>
<td>0.1 0.1 0.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F9</td>
<td>2</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F10</td>
<td>2</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F11</td>
<td>2</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F12</td>
<td>2</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F13</td>
<td>2</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F14</td>
<td>2</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F15</td>
<td>2</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F16</td>
<td>2</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F17</td>
<td>2</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F18</td>
<td>2</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F19</td>
<td>2</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F20</td>
<td>2</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F21</td>
<td>2</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F22</td>
<td>2</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F23</td>
<td>2</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F24</td>
<td>2</td>
<td>0.0 0.0 0.0</td>
<td>0.0 0.0 0.0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Complimentary Contributor Copy
For problems F2~F4, SA-SAES(μ+λ) and SA-SACEP perform similarly. For problems F5~F18, all these two hybrid methods seem to have the same performance too. For problems F19~F22, SA-SAES(μ+λ) and SA-SACEP perform well. For F23~F26, SA-SAES(μ+λ) and SA-SACEP perform the same. For the problems remained, we cannot make some comparison for all the two hybrid methods of this Subsection. However, all the two hybrid methods in this Subsection are better than without hybrids.
4. APPLYING SA TO MOLECULAR MODELING

In [62], Zhang (2011) used the follow SA procedure: “The solvated proteins were then quickly heated from 0 K to 300 K linearly for 20 ps. The systems were kept at 300 K for 80 ps. The systems then were slowly cooled from 300 K to 100 K linearly for 400 ps. The systems were kept for 100 ps at 100 K. All the systems were in constant NVT ensembles using Langevin thermostat algorithm with weak restraints (a force constant of 10.0 kcal mol$^{-1}$ Å$^{-2}$ was used) on the solvated proteins. The SHAKE and SANDER (simulated annealing with NMR-derived energy restraints) algorithms with nonbonded cutoffs of 9 Å were used during the heating, cooling and the 100 ps at 100 K. The equilibration was done in constant NPT ensembles under a Langevin thermostat for 4,400 ps and the RMSD, PRESS, and VOLUME (DENSITY) were sufficiently stable for each model; the jump in RMSD of around 0.2 Å correlates with removing restraints for the change from NVT to NPT, but it did not change the structures at 100 K. Equilibration was under constant pressure 1 atm and constant temperature (100 K) in a neutral pH environment (equilibration was performed at the low temperature of 100 K in order to be consistent with the experimental work). A step size of 2 fs was used for all SA runs. The structures were saved to file every 100 steps. During the SA, the Metropolis criterion was used.” This is an ordinary SA method for molecular modeling and refinement of 100 K crystal structures. Before or after the SA procedure is always using hybrids with local search optimization methods (for example the steepest descent method or the conjugate gradient method or both) in Amber and Gromacs computational chemistry packages.

CONCLUSION

SA is a popular method used in mathematical optimization computations. This article introduced the detailed implementations of SA in optimization, and presented two kinds of hybrids with local and global optimization search algorithms respectively, and then introduced the situation in the use of SA in molecular modeling in computational chemistry crystal structures. The SA theory presented in this article should be very useful for the practical SA calculations.
ACKNOWLEDGMENTS

Dr Zhang JP thanks Professors Bagirov AM and Abbass HA for their computer codes of DG method and evolutionary algorithms.

Participant

Dr Jiapu Zhang.

REFERENCES


Complimentary Contributor Copy


Complimentary Contributor Copy


Complimentary Contributor Copy


Complimentary Contributor Copy


[57] Kaj Madsen (Technical University of Denmark)’s Test problems for global optimization (accessed August 31, 2013): http://www2.imm.dtu.dk/~kajm/Test_ex_forms/test_ex.html


Chapter 2

HANDLING EQUALITY AND INEQUALITY LINEAR CONSTRAINTS IN SIMULATED ANNEALING

Danilo R. Velis*
Universidad Nacional de La Plata,
Facultad de Ciencias Astronómicas y Geofísicas, Argentina;
and CONICET, Argentina

Abstract

Simulated annealing (SA) has become an algorithm of choice when dealing with nonlinear global optimization problems, specially when the objective function is ill-behaved or multimodal, and when derivative information is not available or difficult to evaluate. In general, most SA algorithms are devised for unconstrained optimization, though the use of lower and upper bounds to delimit the feasible region is a common practice. When additional constraints such as equality or inequality constraints are to be taken into account, the development of appropriate strategies to force the solution to satisfy those constraints is critical. In this sense, and for the sake of efficiency, the exploration of the model space to evaluate new and unknown areas within the feasibly region is a

*E-mail address: velis@fcaglp.unlp.edu.ar

Complimentary Contributor Copy
key issue that requires further investigation. This work presents a simple strategy for handling arbitrary linear constraints in SA or other "random search" optimization technique. Both equality and inequality linear constraints are taken into account. In the case of inequality constraints, the optimization problem is reformulated as an unconstrained optimization one with dynamically adapted bounding constraints, which are easily implemented in SA algorithms. In the case of equality constraints, the optimization problem is also transformed into an unconstrained optimization problem with bounding constraints, but, instead, the original unknowns are expressed in terms of a new set of variables through the singular value decomposition (SVD) of the constraint matrix. The key of the proposed technique lies in the fact that, once an initial point within the feasible region is provided, new points are generated by adding random perturbations that lie in the null space at each annealing iteration. Hence, feasibility is maintained at every stage of the SA search process, avoiding unnecessary evaluations of the objective function due to constraints violations. Further, the proposed technique is general and easy to use, requiring neither modifications nor any parameter tuning to work in different problems, making it an interesting alternative to other constraint-handling techniques available in the literature. The strategy is illustrated using SA to minimize testing functions that contain both bounding and equality or inequality linear constraints, including the classical loaded die problem.

Keywords: Simulated annealing, linear constraints, feasible region, interior point, singular value decomposition

1. Introduction

There are several methods for handling constraints in nonlinear optimization, most of them based on local search techniques [Scales, 1985]. When the objective function is highly nonlinear and/or non-differentiable, and when there are several local minima that prevent global convergence, it is well known that local search techniques are prone to fail [Weise, 2008]. In this context, methods such as simulated annealing (SA) and evolutionary algorithms play a key role.

The evolutionary computation community has developed an interesting battery of techniques for dealing with both linear and nonlinear constraints [Coello,
1999, Koziel and Michalewicz, 1999, e.g.]. In the particular case of linear constraints only, GENOCOP (Genetic algorithm for Numerical Optimization for Constrained Problems), a widely used evolution program, is perhaps the most representative [Michalewicz and Janikow, 1991, Michalewicz, 1996]. In this approach, linear equalities are first eliminated in terms of a subset of the original variables, and linear inequalities are updated accordingly. Then, specialized genetic operators are used to generate feasible points at each generation, provided the process is initiated with a feasible point. This is a rather complex strategy, and the reader is referred to the aforementioned works for more details.

In SA optimization, a method based on the theory of discrete Lagrange multipliers has also been proposed [Wah and Wang, 1999, Wah and Wang, 2000]. The method, which is called constrained simulated annealing (CSA), is able to cope with both linear and nonlinear constraints simultaneously by introducing an additional search space for the Lagrange multipliers. Here, feasibility is not maintained at all stages of the iterative process, but strong convergence criteria are defined that guarantee convergence in annealing time. The results showed in these works demonstrate that CSA outperforms some evolutionary algorithms and local-search methods when using several test problems available in the literature. It is important to stress that in CSA, feasibility is not preserved but at convergence, thus the quality of the results may be limited in problems with a large number of constraints.

Other stochastic methods include multistart procedures which are based on greedy searches that start from different initial points. The global minimizer is then selected among all local solutions after a number of searches have been performed. This is the case of DONLP2, a popular sequential quadratic programming package used in constraint optimization [Spellucci, 1998]. The success of these strategies relies, as expected, on the starting points.

Here I propose simple strategies aimed to maintain the feasibility throughout the whole iterative process when dealing with linearly constrained global optimization problems. The goal is to focus on minimizing the objective function, honoring all the constraints at every stage of the iterative process using simple algebraic procedures. As a result, rather than having to drive a potential solution that lies in the unfeasible region towards the feasible region, as in the case of, for example, the methods based on penalty functions, all the computational effort is put into minimizing the objective function only. In the proposed strategies, first,
equality linear constraints are automatically eliminated and the original model space (unknowns) are replaced by a new set of variables with dynamic bounds, which are easily handled by SA algorithms. Then, the original inequality constraints, which define a convex space, are rewritten in terms of the new variables and incorporated into an augmented set of bounding constraints for the new variables. As a result, the general linearly constrained optimization problem, which may contain equality and/or inequality (including bounding) constraints, is automatically transformed into a constrained optimization problem with bounding constraints only. These bounding constraints represent dynamic search ranges in the SA iterative process that need to be updated at every iteration.

The two numerical examples that illustrate the proposed method show the high efficiency and accuracy that is expected to be obtained when dealing with any type of linear equality and/or inequality constraints in the context of SA optimization. The first example is a two dimensional nonlinear function with inequality constraints only, whose global minima are located on the vertex of the feasible region. The other example is the classical (loaded) die problem, that can be solved using the maximum entropy method [Jaynes, 1982]. In this case, both equality and inequality constraints are included.

2. Theory and Methods

2.1. Problem Statement

The global nonlinear constrained optimization problem consists in finding the absolute minimum value of the function

\[ f = f(x), \quad x \in X, \quad (1) \]

where \( x = (x_1, x_2, \cdots, x_m)^T \) is the vector of \( m \) unknowns, and \( X \) is the model space of dimension \( m \), subject to

\[ g_j(x) \leq 0, \quad j = 1, \cdots, q \quad (2) \]

and

\[ h_j(x) = 0, \quad j = 1, \cdots, p, \quad p < m. \quad (3) \]
The set of points \( x \in X \) that satisfy the constraints (2) and (3) comprises the \textbf{feasible region} \( \mathcal{F} \). The solution to the optimization problem is denoted by \( x^* \), \( x^* \in \mathcal{F} \), such that \( f(x^*) \leq f(x), \forall x \in X \cap \mathcal{F} \). If there are no exact constraints and all inequality constraints are linear, \( \mathcal{F} \) is a \textit{convex space}. Further, if the objective function is also linear, \textit{linear programming} can be used to find the absolute minimum, which corresponds to a vertex of the convex space. \textit{Quadratic programming} can be used whenever the objective function is quadratic and the constraints are linear. In general, \textit{nonlinear programming} refers to the problem where at least one of the expressions is nonlinear. Here I will consider the case where the nonlinearity is contained in the objective function only. A description of how feasible points are generated in a SA optimization scenario follows. For this purpose, equality and inequality linear constraints are treated separately.

### 2.2. Handling Equality Linear Constraints

The problem can be written as

\[
\begin{align*}
\text{minimize} & \quad f(x) \quad \text{subject to} \quad Ax = b, \\
\end{align*}
\]

where \( A \) and \( b \) are a \( p \times m \) matrix and a \( p \times 1 \) column vector, respectively, that define the equality constraints (3). The solution is to be found in the domain \( X \), \( x_i \in [\underline{x}_i, \bar{x}_i], i = 1, \cdots, m \).

Let \( x_0 \) be a particular solution (feasible point) of \( Ax = b \). Since \( p < m \), this is an under-determined system of linear equations that can be solved, for example, using the singular value decomposition (SVD) [Press et al., 1992]. The purpose here is to generate a new feasible point by “randomly” perturbing the initial point. To this end, let \( \delta x \) be a perturbation that produces a new solution:

\[
\begin{align*}
x = x_0 + \delta x, \quad x \in \mathcal{F}. \\
\end{align*}
\]

Replacing expression (5) into the equality constraints (3), I obtain

\[
\begin{align*}
A\delta x = 0. \\
\end{align*}
\]

This result suggests that in order to obtain a new feasible point in the model space, it is necessary to generate perturbations \( \delta x \) which are solutions to the homogeneous system of linear equations (6). In a SA process, \( \delta x \) should be as
random as possible, not to bias the solution towards a particular subregion of \( F \). Moreover, the entire feasible region should be allowed to be explored during the SA iteration. Hence, it is desirable to devise a perturbation strategy that allows to obtain all the solutions to equation (6). For this purpose, I write the SVD of matrix \( A \):

\[
A = U \Sigma V^T, \tag{7}
\]

where \( U \) and \( V \) are \( p \times p \) and \( m \times m \) square matrices containing the left and right eigenvectors, respectively, and \( \Sigma \) is a \( p \times m \) matrix of rank \( p \) containing the singular values of \( A \). Since \( p < m \), there are at least \( m - p \) singular values that are identically zero. This means that any column of \( V \) for which the corresponding singular value is zero, is a solution of equation (6). Moreover, any linear combination of the columns of \( V \) whose corresponding singular values are zero, span the entire null space and produce solutions to equation (6) [Press et al., 1992]. In other words, the entire feasible domain \( F \) is spanned. As a result, the perturbation \( \delta x \) can be written as

\[
\delta x = \tilde{V} y, \tag{8}
\]

where \( \tilde{V} \) is the \( m \times (m - p) \) matrix containing the aforementioned columns of \( V \), and \( y \) is the \( m - p \) column vector containing new variables to be determined. Then, the optimization problem (4) can be rewritten as a bounding constrained optimization problem where the unknowns are the elements of \( y \):

\[
\text{minimize} \quad f(x) = f(x_0 + \tilde{V} y) = \tilde{f}(y), \tag{9}
\]

on the new domain \( \tilde{F} \), which is defined by the constraint \( x + \tilde{V} y \in X \). So, it remains to determine the new bounds in the search space of the unknowns \( y \). A first approximation can be obtained by considering the fact that \( V \) is orthogonal, therefore

\[
||\delta x||^2 = \delta x^T \delta x = y^T \tilde{V}^T \tilde{V} y = y^T y = ||y||^2. \tag{10}
\]

Thus, given a starting point \( x_0 \), the maximum perturbation \( ||\delta x|| \) can be determined according to the limits that define the domain \( X \). This value should be high enough to permit the system to reach any point in the domain \( X \). Then, in
order to generate feasible points within this region, \( y \) must lie within the hypersphere of radius \( ||\delta x|| \), as denoted by equation (10).

However, and for the sake of generality, it is possible that during the iterative process involved in the SA scheme, models \( x \) lying outside the domain \( X \) are generated. To avoid this, every time a configuration \( y \) is generated, the new point \( x \) is checked whether it belongs to \( X \). If it does not, the generation process can be repeated until \( x \in X \), or better, the perturbation can be scaled to produce a new perturbation

\[
\delta x' = \alpha \delta x,
\]

where \( \alpha \) is a scale factor obtained by determining the maximum deviation from the bounds imposed by the original domain \( X \):

\[
\alpha = \frac{1}{||\delta x||} \min \left\{ \max_i \{ |x_i - \bar{x}_i| \}, \max_i \{ |\bar{x}_i - \underline{x}_i| \} \right\}, \quad i = 1, \ldots, m. \quad (12)
\]

Equations (11) and (12) should be applied only when \( x_0 + \delta x \notin X \). As a result, it is always true that \( x_i = x_0_i + \delta x'_i \in [\bar{x}_i, \underline{x}_i], \forall i \).

An alternative procedure is to use dynamic bounds for generating new points \( y \), so that \( x \in X \) at every stage of the SA iteration. Combining equations (5) and (8), and considering the domain \( X \), the dynamic bounds associated with \( y \), \( [\bar{y}_j, \underline{y}_j] \), can be written as

\[
\left[ \max_{i, \bar{y}_j \neq 0} \left\{ \frac{X_j - x_{0i} - y_{ij}}{\bar{v}_ij}\right\}, \min_{i, \underline{y}_j \neq 0} \left\{ \frac{x_i - x_{0i} - y_{ij}}{\bar{v}_ij}\right\} \right], \quad j = 1, \ldots, m - p; \quad (13)
\]

where

\[
y_{ij} = \sum_{k=1, k \neq j}^{m-p} \bar{v}_{ijk} y_k. \quad (14)
\]

The bounds should be updated after each element \( y_j \) is generated, to ensure that \( x_i \in [\bar{x}_i, \underline{x}_i] \) at every iteration.
2.3. Handling Inequality Linear Constraints

In this case, the problem can be written as

\[
\text{minimize } f(x) \text{ subject to } Cx \leq d,
\]

(15)

where \( C \) and \( d \) are the \( q \times m \) matrix and the \( q \times 1 \) column vector that define the inequality constraints (2). The domain \( X \) can be included as additional inequality constraints in the above formulation. Thus, the feasible domain \( F \) is now given by the convex space determined by the set of inequalities.

As stated before, the philosophy here is to start with an initial point \( x_0 \) within the feasible region, and generate new points \( x = x_0 + \delta x \) that also lie within the feasible region. In this sense, Berbee et al. (1987) presented a “hit-and-run” (HAR) algorithm to generate interior points with uniform distribution. Chang and Hsieh (1995) applied the technique in connection to color-signal separation using SA. The key of the procedure is to start with an interior point, \( x = x_0 \), and to generate a random direction \( u \). The line \( x + \gamma u \) is defined, where \( \gamma \) is a line parameter that determines the size of the perturbation \( \delta x \). Then, the two hit-points where the line intercepts the convex domain are computed. The new interior point is taken randomly from the line segment so determined.

The algorithm proceeds as follows:

1. Start with an interior point \( x = x_0 \).
2. Generate a random unit vector \( u \) from the \( m \) possible coordinate vectors and their opposite vectors.
3. Determine:
   \[
   \gamma_j = (d_j - c_j^T x) / (c_j^T u), \quad j = 1, \cdots, q;
   \]
   \[
   \gamma^+ = \min_j [\gamma_j | \gamma_j > 0];
   \]
   \[
   \gamma^- = \max_j [\gamma_j | \gamma_j < 0],
   \]
   where \( c_j^T \) and \( d_j \) are the \( j \)th row and \( j \)th entry of \( C \) and \( d \), respectively.
4. Generate the new interior point \( x' = x + [\gamma^- + \theta(\gamma^+ - \gamma^-)] u \), where \( \theta \) is a random variable sampled from a uniform distribution on \([0, 1]\).
The parameters $\gamma^+$ and $\gamma^-$ determine the dynamic bounds for the new point $x'$ in the direction $u$. That is

$$x'_i \in [x_i + \gamma^-, x_i + \gamma^+], \quad i = 1, \cdots, m$$  \hfill (16)

where the HAR process described above is repeated for each dimension until all bounds and coordinates are updated. In essence, this is equivalent to the alternative scheme described in the previous section regarding the dynamic ranges for the parameters $y_i$’s in the exact linear constraints case, as denoted by equation (13).

Unlike the original HAR algorithm described by Berbee et al. (1987), who rely on a uniform distribution to generate new interior points, $\theta$ could actually be sampled from any other distribution (e.g., Gaussian, Cauchy, etc.), provided the new point does not fall outside the feasible region. For this purpose, instead of generating new interior points as stated in the step (4), I prefer to generate them using

$$x' = \begin{cases} 
    x + \theta \gamma^- u & \text{if } \theta < 0 \\
    x + \theta \gamma^+ u & \text{if } \theta \geq 0,
\end{cases}$$  \hfill (17)

where $\theta$ is a random variable on $[-1, 1]$ drawn from a non-uniform distribution (e.g., Cauchy-like), whose “dispersion” decreases with temperature. This is important because it is well known that SA algorithms using other than a uniform distribution to generate new states are much more efficient [Ingber, 1989, Ingber, 1993, Ingber, 1996].

Finally, note that at each annealing iteration there are two possibilities of generating and accepting/rejecting the new points according to the value of the cost function:

1. apply the HAR algorithm once per annealing iteration and evaluate the cost function,
2. repeat the HAR algorithm until the $m$ dimensions are perturbed before evaluating the cost function. In this case, the $m$ dimensions can be selected one at a time in an ordered sequence (the strategy I used in the numerical examples), or they can be put in a random sequence predefined at every SA iteration.

Complimentary Contributor Copy
2.4. Handling both Equality and Inequality Linear Constraints Simultaneously

Both strategies described in the previous two sections can be combined into a single strategy that takes into account all types of linear constraints. Actually, when describing the equality linear constraints case, I considered bounding constraints as well, which are a special case of inequality constraints. But for the sake of generality, I will derive the expressions that can be used for the case where the optimization problem includes both equality and inequality (including bounding) constraints simultaneously.

First, notice that the case with equality linear constraints has been transformed into a problem with bounding constraints only, as denoted by equation (13). This was possible because the $p$ constraints on the unknown vector $x$ have been eliminated through the use of the SVD, leading to a problem with bounding constraints on the new unknown vector $y$. Then, since bounding constraints can be treated like ordinary inequality constraints, all we have to do is to re-define the HAR algorithm in terms of the new unknown vector $y$. To this end, I combine equations (5) and (8), and taking into account the bounding constraints for $x$, I get

$$C_b y \leq d_b,$$

where $C_b$ and $d_b$ are a $2m \times (m - p)$ block matrix and a $2m \times 1$ column vector defined by

$$C_b = \left( \begin{array}{c} \tilde{V} \\ -\tilde{V} \end{array} \right) \quad \text{and} \quad d_b = \left( \begin{array}{c} \bar{x} - x_0 \\ -\bar{x} + x_0 \end{array} \right),$$

respectively. Note that in the case that $p = 0$ (i.e. no equality constraints), one can set $y = x$ and $\tilde{V} = I$, where $I$ is the $m \times m$ identity matrix. Thus, equations (18) and (19) can be used to handle both equality and bounding constraints irrespectively of the value of $p$.

It remains to set the appropriate inequality constraints $Cx \leq d$ in terms of the unknown variable $y$. Again, combining equations (5) and (8), and taking into account the inequality constraints, I can write

$$C_q y \leq d_q,$$
where \( C_q \) and \( d_q \) are a \( q \times (m-p) \) matrix and a \( q \times 1 \) column vector defined by

\[
C_q = C \tilde{V} \quad \text{and} \quad d_q = d - Cx_0 ,
\]

respectively. Finally, combining equations (18) and (20), I get

\[
Gy \leq h , \quad (22)
\]

where \( G \) and \( h \) are a \((2m+q) \times (m-p)\) block matrix and a \((2m+q) \times 1\) column vector of the form

\[
G = \begin{pmatrix}
\tilde{V} \\
-\tilde{V} \\
CV
\end{pmatrix} \quad \text{and} \quad h = \begin{pmatrix}
\bar{x} - x_0 \\
-x + x_0 \\
d - Cx_0
\end{pmatrix} ,
\]

respectively. Hence, equations (22) and (23) take into account all equality and inequality (including bounding) constraints on the unknown variable \( x \), but written in terms of the new unknown variable \( y \). As a consequence, the general linearly constrained optimization problem defined in Section 2.1. has been replaced by the simpler problem

\[
\begin{align*}
\text{minimize} & \quad f(y) = f(x_0 + \bar{V}y) \\
\text{subject to} & \quad Gy \leq h , \quad (24)
\end{align*}
\]

which is equivalent to the problem defined in (15). This means that, given an initial feasible point \( x_0 \), the general optimization problem containing both equality and inequality linear constraints can be solved with the aid of the HAR algorithm to generate new feasible points at every stage of the SA iterative process.

### 3. Numerical Examples

The first example takes into account only inequality linear constraints. It is a two-dimensional function defined by

\[
f(x) = \begin{cases} 
  x_2 + 10^{-5} (x_2 - x_1)^2 - 1 , & 0 \leq x_1 < 2 , \\
  \frac{1}{27 \sqrt{3}} ((x_1 - 3)^2 - 9)x_2^3 , & 2 \leq x_1 < 4 , \\
  \frac{1}{3} (x_1 - 2)^3 + x_2 - \frac{11}{3} , & 4 \leq x_1 \leq 6 
\end{cases} \quad (25)
\]
subject to
\[ \begin{cases} \frac{x_1}{\sqrt{3}} - x_2 \geq 0, \\ -x_1 - \sqrt{3}x_2 + 6 \geq 0, \\ 0 \leq x_1 \leq 6, \\ x_2 \geq 0. \end{cases} \] (26)

This function, which is depicted in Figure 1a, corresponds to the test function 6 in Ji et al. (2006). There are three global minima at \( x^* = (0, 0), (3, \sqrt{3}) \) and \( (4, 0) \), where \( f(x^*) = -1 \). For the minimization, I used Very Fast Simulated Annealing (VFSA) [Ingber, 1989] with \( x_0 = (3, \frac{1}{2}) \) and a maximum number of iterations of 1000, where each SA iteration involves a single evaluation of the function. Figure 1b shows all the generated and accepted models for two independent SA runs that converged to two different global minima. Note that all points belong to the feasible region. This contributes to accelerate the convergence and reduce the number of function evaluations because the focus is put on minimizing \( f(x) \) only. Figure 2 shows the convergence curves for 100 independent SA runs. In most cases, the convergence is achieved before 400–600 iterations with a high accuracy, as shown in Table 1. The table shows the mean function at the global minima after 100 independent SA runs using \( N = 200 \) to 2000, where \( N \) is the number of SA iterations or function evaluations. Note the high accuracy even when the number of function evaluations is relatively small. As compared with the results reported by Ji at al. (2006) (see Table 4 in that article), the performance of the proposed technique is much more efficient. In effect, the method proposed by Ji et al. (2006), which is called “improved SA” (ISA) and is specially devised to deal with linear constraints, obtains one of the global minima (with a moderate accuracy) in about 4700 function evaluations, while GENOCOP [Michalewicz, 1996] requires, as reported by Ji et al. (2006), about 35000.

The next example involves the classical loaded die problem [Jaynes, 1982, e.g.]. The problem consists in estimating the probability (actually the frequency) associated with each face when the die is tossed a given number of times. For a die with \( n \) faces, the solution can be obtained by maximizing the entropy
\[ H(x) = -\sum_{i=1}^{n} x_i \log x_i, \] (27)
Table 1. Test function (25) example: mean function at the global minimum (and associated standard deviations $\sigma$) after 100 independent SA runs. $N$ stands for the number of SA iterations or function evaluations.

<table>
<thead>
<tr>
<th>$N$</th>
<th>$f(x^*)$</th>
<th>$\sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>-0.9972114</td>
<td>$0.73 \times 10^{-2}$</td>
</tr>
<tr>
<td>400</td>
<td>-0.9997948</td>
<td>$0.65 \times 10^{-3}$</td>
</tr>
<tr>
<td>600</td>
<td>-0.9999597</td>
<td>$0.18 \times 10^{-3}$</td>
</tr>
<tr>
<td>800</td>
<td>-0.9999979</td>
<td>$0.62 \times 10^{-5}$</td>
</tr>
<tr>
<td>1000</td>
<td>-0.9999988</td>
<td>$0.58 \times 10^{-5}$</td>
</tr>
<tr>
<td>2000</td>
<td>-1.0000000</td>
<td>$0.26 \times 10^{-7}$</td>
</tr>
</tbody>
</table>

Figure 1. a) Test function (25) with inequality constraints. b) Generated (crosses) and accepted (squares) models during two independent SA runs that converged to two different global minima (1000 iterations).
Figure 2. Convergence curves for 100 independent SA runs corresponding to
the minimization of the test function \( f(x) \) shown in Figure 1.

subject to

\[
\left\{ \begin{array}{l}
\sum_{i=1}^{n} x_i = 1, \\
0 \leq x_i \leq 1, \quad i = 1, \cdots, n
\end{array} \right.
\] (28)

This constrained optimization problem involves both equality and inequality linear constraints. When there is no reason to think that any face is more likely than any other, or, in other words, when we are not given any information except that there are \( n \) faces, then the intuition indicates that the probability associated with each face is the uniform distribution \( x_i = 1/n, i = 1, \cdots, n \). Actually, this is the solution that is obtained when equation (27) is maximized, without the need to perform any empirical experiment that involves tossing the die a given number of times. For \( n = 6, x_i = \frac{1}{6} \) and \( H(x) = \log 6 = 1.7917595 \). Further, the average number of spots up, that can be computed using

\[
\sum_{i=1}^{6} ix_i = w_i,
\] (29)

is equal to 3.5.

Now suppose that the die is loaded, that is, we have new evidence that indicates that the average number of spots up is not 3.5. Clearly, the solution will depart from the uniform distribution. So, what are the probabilities associated with each face? The answer to this question can be obtained, again, using the
Handling Equality and Inequality Linear Constraints ...

Table 2. Die problem example: mean entropy at the global minimum (and associated standard deviations $\sigma$) after 100 independent SA runs

<table>
<thead>
<tr>
<th>$N$</th>
<th>$w = 3.5$</th>
<th>$w = 4.5$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$H(x^*)$</td>
<td>$\sigma$</td>
</tr>
<tr>
<td>200</td>
<td>1.7916501</td>
<td>$0.13 \times 10^{-3}$</td>
</tr>
<tr>
<td>400</td>
<td>1.7917567</td>
<td>$0.37 \times 10^{-5}$</td>
</tr>
<tr>
<td>600</td>
<td>1.7917592</td>
<td>$0.31 \times 10^{-6}$</td>
</tr>
<tr>
<td>800</td>
<td>1.7917594</td>
<td>$0.29 \times 10^{-7}$</td>
</tr>
<tr>
<td>1000</td>
<td>1.7917595</td>
<td>$0.26 \times 10^{-7}$</td>
</tr>
<tr>
<td>2000</td>
<td>1.7917595</td>
<td>$0.77 \times 10^{-8}$</td>
</tr>
</tbody>
</table>

maximum entropy method adding the additional equality constraint (29) with $w \neq 3.5$. For $w = 4.5$, for example, the solution is given by

$$x_i = \frac{e^{-\lambda_i}}{\sum_i e^{-\lambda_i}}, \quad i = 1, \cdots, 6 \quad (30)$$

with $\lambda = -0.37105$. So, $x^* = (0.0543, 0.0788, 0.1142, 0.1654, 0.2398, 0.3475)$, and $H(x^*) = 1.6135811$ [Jaynes, 1982].

I now illustrate the behavior of VFSA together with the proposed strategy for handling both equality and inequality constraints simultaneously, to minimize $f(x) = -H(x)$ subject to the constraints (28) and (29), for $w = 3.5$ and $w = 4.5$. In the first case, I set $x_0 = (0.1, 0.3, 0.1, 0.1, 0.3, 0.1)$, while for the loaded die, I set $x_0 = (0.1, 0.1, 0.1, 0.1, 0.1, 0.5)$. After a few hundred iterations (function evaluations), the solutions are obtained with a high accuracy, as shown in Tables 2 and 3.

For the sake of simplicity, the corresponding standard deviations associated with $x^*$ are not shown. Instead, Figure 3 shows the convergence curves for 100 SA runs (Figure 3a), and the evolution of the six unknowns for a single SA run (Figure 3b), for the case of the loaded die with $w = 4.5$. 

Complimentary Contributor Copy
Table 3. Die problem example: mean solutions after 100 independent SA runs

<table>
<thead>
<tr>
<th></th>
<th>$w = 3.5$</th>
<th></th>
<th>$w = 4.5$</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$x^*$</td>
<td>$N = 200$</td>
<td>$N = 600$</td>
<td>$N = 200$</td>
<td>$N = 600$</td>
</tr>
<tr>
<td>$x_1$</td>
<td>0.1670</td>
<td>0.1667</td>
<td>0.0544</td>
<td>0.0544</td>
</tr>
<tr>
<td>$x_2$</td>
<td>0.1664</td>
<td>0.1667</td>
<td>0.0785</td>
<td>0.0788</td>
</tr>
<tr>
<td>$x_3$</td>
<td>0.1663</td>
<td>0.1667</td>
<td>0.1144</td>
<td>0.1142</td>
</tr>
<tr>
<td>$x_4$</td>
<td>0.1668</td>
<td>0.1667</td>
<td>0.1653</td>
<td>0.1654</td>
</tr>
<tr>
<td>$x_5$</td>
<td>0.1670</td>
<td>0.1667</td>
<td>0.2400</td>
<td>0.2398</td>
</tr>
<tr>
<td>$x_6$</td>
<td>0.1665</td>
<td>0.1667</td>
<td>0.3473</td>
<td>0.3475</td>
</tr>
</tbody>
</table>

Figure 3. (Loaded) die problem example: a) convergence curves for 100 independent SA runs, and (b) convergence curves for the six unknowns for a single SA run.
4. Conclusion

I have presented and described various strategies for dealing with equality and inequality linear constraints in global optimization algorithms such as SA. Whether the original problem contains only equality linear constraints, or both, I showed how any linearly constrained optimization problem can be transformed into an optimization problem with bounding constraints only. In a practical context, this result is very convenient because bounding constraints need not any special treatment and are easily (and naturally) incorporated into any SA algorithm. The only caveat is that these bounding constraints, which represent search ranges, should be updated dynamically at every model perturbation throughout the SA iteration. However, the processes associated with this search ranges updating are very simple and rely on a few algebraic operations that do not represent any significant computational complexity. Further, all operations can be incorporated automatically within a general SA code, without any special tuning of parameters or problem reformulation. One limitation of the proposed strategies may lie in the fact that a feasible point (i.e. a point that satisfies all the constraints) is required to start the iteration. In the case of exact linear constraints this step does not represent any difficulty, because methods such as the SVD could be used to easily obtain that point. In the case of inequality linear constraints, other strategy needs to be applied to guarantee that a feasible point is available at the beginning of the iterative process. In any case, what is important to stress here is that once an initial point is provided, all model perturbations are guaranteed to lead to another feasible point, spanning the whole feasible region randomly using either a uniform or any other probability distribution function, as shown for the VFSA case. The numerical examples show that the strategy permits to attain a high efficiency and accuracy, because all computational effort is put into minimizing the cost function without wasting time on function evaluations trying to drive the solution to the feasible region.

References


Chapter 3

**AIRCRAFT DESIGN WITH SIMULATED ANNEALING FOR SAE AERODESIGN COMPETITION**

*Diego Movio Borburema¹, Marcos de Sales Guerra Tsuzuki¹, Antonio Luis de Campos Mariani¹, Andre Kubagawa Sato¹, and Thiago de Castro Martins¹*

¹ Computational Geometry Laboratory, Escola Politécnica da Universidade de São Paulo, São Paulo, SP, Brazil

**Abstract**

The SAE AeroDesign competition aims to reward the aircraft with the best design and with the best performance. The performance is measured as the ratio between the lifted weight and the aircraft weight. This competition has grown in the last few years, as well the teams’ competitiveness; they are getting very close to reaching the perfect aircraft configuration. This research aims at determining an efficient methodology to get a competitive aircraft design and performance. Simulated annealing is applied as an optimization algorithm to determine such aircraft design. It is shown that the aircraft design can be represented by a parametric model. Before

---

*E-mail address: diego.movio@gmail.com, mtsuzuki@usp.br, camposmariani@gmail.com, thiago@usp.br, andre.kubagawa@usp.br*
evaluating the aircraft design performance, it is necessary to check if the aircraft design obeys the competition constraints. If the constraints are not obeyed the aircraft design is invalid. Several parametric schemes can be used to represent the aircraft design. However, a key point is defining one parameter scheme in which few parametric models are invalid. Thus way, the domain search by the simulated annealing is more effective. An effective parametric scheme is proposed here and a simulated annealing with crystallization heuristic is used to determine an optimal aircraft design. Several results are shown and compared with other parametric schemes as well with Genetic Algorithm.

**Keywords:** Parametric representation scheme, simulated annealing, constraint optimization

1. Introduction

The Society of Automotive Engineers (SAE) Brazil holds an annual competition event, AeroDesign, in which engineering students are challenged to organize, design and build a radio-controlled aircraft to carry as much weight as possible. These problems highlight many real challenges faced in aerospace engineering.

Each competing team builds a stable aircraft so as to achieve the best score. The regulation of the competition specifies a series of restrictions, related to the geometry (dimensional sum), resources (motor type, loading and fuel) and runway length, applied for both lifting and landing.

An aircraft score is determined by a list of factors such as reliability, determined by using an accuracy curve that shows the best performance of the aircraft given temperature and pressure conditions, total load (sum of weight of the aircraft and the lifted weight), and structural efficiency (ratio between the lifted weight and the aircraft’s weight). Therefore, there is an incentive to build light, structural efficient aircrafts. For this, it is necessary to create a conceptual project which aims at the highest score, but also respects the restrictions.

Raymer [8] researched the aircraft conceptual design process and used genetic algorithms to improve it. The best results derived from a genetic algorithm evolutionary scheme called Killer Queen. The results obtained from the algorithm proposed, Simulated annealing with crystallization heuristic, has been applied to solve several types of problems with different types of parameters.
(float, integer and cyclic) and different types of objective functions (discrete and continuous). In our research, simulated annealing is used to determine the best aircraft conceptual design. Section 2 explains the project constraints and the aircraft score evaluation flowchart. Section 3 explains the importance of the representation scheme and its relation with the constraints. Section 4 explains the simulated annealing with crystallization heuristic used. Section 5 shows some results and compares our proposed method with the Killer Queen used by Raymer [8]. Section 6 shows the constructed aircraft, followed by the conclusions in section 7.

2. Aircraft Project Evaluation

Aircraft designs defined by parametric models need to be evaluated in order to determine their validity and score. A generic aircraft design is shown in Fig. 1. Validation is defined by AeroDesign regulation restrictions, as is the project score.

Figure 1. Generic aircraft project. Wing, fuselage and tail are illustrated.

The flowchart in Fig. 2 shows the steps for the validation and scoring of an aircraft design which follows the Aerodesign regulation. The constraints are verified following the sequence shown. When a constraint is violated, the aircraft design is considered invalid and its score is set to zero. The aircraft score is only determined if all the constraints have been observed. The constraints are: dimensional limit, aerodynamics, stability and structures.
Figure 2. Flowchart for the evaluation of the aircraft design validity and score.

The first evaluation is the dimensional limit, which checks that the sum of height, width and length of the aircraft model is below a fixed limit. Aerodynamic evaluation is verified next and depends upon the wing and tail geometry. The stability of the aircraft is checked in the next step. In this module, the approximate weight of the aircraft is determined using the material density and the volume of the stringer. Motor placement is decided based on the aircraft center of gravity. However, it must be placed more than 8 cm away from the fuselage, otherwise the design is considered invalid.

Finally, if every constraint is satisfied, the score is determined based on the results of the carried weight capacity calculations, considering the runway length limit.
3. Representation Schemes

Fig. 3 shows an example of a representation scheme [4] which maps parametric models (on the left) to aircraft projects (on the right). The dark region on the left corresponds to the set of parametric models which are not associated to any aircraft model and the darker region on the right is the set of parametric models which cannot be represented by any parametric models.

![Diagram showing representation scheme cases](image)

Figure 3. Example of a representation scheme. The parametric space is located on the left and the aircraft designs are located on the right.

Ideally, the representation scheme adopted should contain no darker regions. In this case, every parametric model is valid and each aircraft project can be represented parametrically. Once the parameters are defined, the solution can be directly obtained, no additional evaluation would be necessary to validate a solution.

Four mapping examples are shown in Fig. 3. Case 1 represents the mapping of a parametric model to a feasible aircraft model. The second case also represents a valid mapping, in which one aircraft design can be represented by two parametric models. An invalid mapping occurs in case 3, in which one parametric model corresponds to two aircraft designs. In this situation, there is an ambiguity as it is not possible to distinguish between both aircraft designs using the representation scheme adopted.
3.1. First Attempt

Fig. 4 shows the measurements used to define the aircraft geometry. A first approach was to use these values as parameters. After some tests, it was possible to observe that there were a large number of invalid parametric models, indicating that the invalid region in the parametric space was very large. Thus, using this representation scheme, it is very hard to perform the domain space exploration.

Using randomly generated parameters, the aircraft designs generated rarely satisfied the dimensional sum restriction, which is the sum of seven dimensions: $m_1, m_2, m_3, m_5, m_6, m_7$ and $m_9$ (see in Fig. 4).

3.2. Approach Proposed

New parametric representation schemes were studied for the airplane model. After many trials, a new scheme was adopted. Three principal measures were adopted: $\alpha$, $\beta$ and $\gamma$, whose definitions are as follows:
• \( \alpha = m_9 \);

• \( \beta = m_1 + m_2 + m_5 + m_7 \) (fuselage distance from the airfoil is constant);

• \( \gamma = m_3 + m_6 \).

These three measures are determined such that the dimensional constraint is always satisfied. First, measure \( m_9 \) is assigned the same value as \( \alpha \), then \( m_1 \) is determined using a specific interval and, finally, \( m_7 \) and \( m_5 \) are determined. Measure \( m_2 \) is the difference between the sum of the calculated measures and \( \beta \). Following a similar procedure, \( m_6 \) and \( m_3 \) are determined according to the value of \( \gamma \).

Dimensions \( m_4 \) and \( m_{10} \) are defined afterwards so as to guarantee that the fuselage meets the lift weight volume limit. \( m_8 \) definition has some flexibility, which allows for a broader search in the aircraft sample space.
4. Simulated Annealing

The simulated annealing [3] is a probabilistic meta-heuristics based on local exploration. This type of optimization heuristic analyzes one, and only one solution for each iteration. Its cost is compared with the previous iteration solution and using heuristics rules, a new solution is generated for the next iteration.

The candidate solution $x_i^*$, evaluated at iteration $i + 1$, is randomly chosen in the neighborhood of the current solution. The current solution $x_i$ may be replaced by the candidate solution $x_i^*$ according to the following rules:

- $f(x_i^*) \leq f(x_i)$: In this case, the current solution is immediately replaced by the new solution.

- $f(x_i^*) > f(x_i)$: the current solution can be replaced with the new solution according to a probability determined by the difference between costs and by the “temperature” of the process.

The “temperature” is a parameter that varies throughout the process according to a “cooling scheduling”. In general terms, the higher the temperature, the more likely it is for a low-cost solution to be replaced by a superior-cost solution (moving the solution in the ascent direction of the cost gradient). This is the strategy which allows the process to escape local minimum. As the process continues, the temperature is gradually reduced, allowing for a stationary optimal solution.

The initial algorithm proposed by Kirkpatrick et al. [3] was applied to combinatorial problems. Corana et al. [1] adapted the simulated annealing to solve problems with continuous parameters. In this adaptation, the concept of Euclidean distance between two possible solutions was introduced. Kirkpatrick et al. [3] had already observed that the simulated annealing has two phases: the exploration and the refinement phases. According to Corana et al. [1], during the exploration phase, the simulated annealing allows the explored neighborhood to be larger, whereas in the refinement phase, the neighborhood is reduced in size. The size of the neighborhood changes according to the temperature in order to keep the number of accepted solutions above at an acceptable level, as rejected solutions candidates do not contribute to the evolution of the solution.

Ingber [2] introduced the ASA (“Adaptive Simulated Annealing”), in which the standard deviation of the probability density function of possible solutions

Complimentary Contributor Copy
reduces with the temperature according to Cauchy’s distribution. Thus, the highest possible value does not reduce with the temperature cooling, but the probability of selecting candidate solutions which are distant from the actual solution increases.

Martins and Tsuzuki [5] proposed a simulated annealing algorithm that automatically determines the most suitable standard deviation for the distribution of probabilities of each parameter during the determination of the next candidate (see Algorithm 1). Hence, each parameter has its own probability distribution, independent of the temperature. One main concept used in this algorithm is the modification of just one parameter per iteration. Following the modification is the evaluation of its contribution to the objective function. If the solution is accepted, it indicates that the standard deviation is relatively small for this parameter and it can be increased ($c_k$ is decreased). Otherwise, it can be reduced. The standard deviation of the probability distribution is controlled by the so-called crystallization factor $c_k$. Each parameter has its own crystallization factor and it dictates the probability distribution of its movements. Its objective is to reduce the amplitude of movement of a parameter which leads to rejected solutions [7].

The simulated annealing with crystallization heuristic was used to solve different types of problems: polygon packing [5] (integer, float and cyclic parameters, with discrete objective function); electrical impedance tomography [6] (float parameters, numerically ill-conditioned objective function); and robot path planning [9] (float parameters and objective function).

5. Discussions and Results

The initial attempt employed the first parametric scheme, which used 10 measurements, and the genetic algorithm to search for the optimal solution, it took up to 25 hours of processing time to obtain the first population of valid solutions. Fig. 6 shows the Killer Queen behavior used (with maximum, average and minimum cost function), in which the first parametric representation scheme had a great influence, as a high number of invalid models were obtained during its execution. Using the new proposed representation scheme, the initial solution (or population) for the meta-heuristic can be obtained in a fraction of that time.

Complimentary Contributor Copy
Algorithm 1: Simulated Annealing with crystallization heuristic.

1. $x \leftarrow \text{random initial solution}$;
2. $T_0 \leftarrow \text{initial temperature}$;
3. while $\text{not finish}$ do
   4. while $\text{global condition is not satisfied}$ do
      5. $T_i \leftarrow T_i \ast \alpha$;
      6. $i \leftarrow i + 1$;
      7. while $\text{local condition is not satisfied}$ do
         8. $k \leftarrow \text{select a parameter to be modified}$;
         9. $x_k^* \leftarrow x_k + \frac{1}{c_i} \sum r(-1, 1) \cdot \Delta r$;
         10. $\Delta E = F(x^*) - F(x)$;
         11. if $\Delta E < 0$ then
             12. $x \leftarrow x^*$;
             13. $c_i \leftarrow c_i - 1$;
         else
            14. if $r(0, 1) < e^{-\Delta E/kT}$ then
                15. $x \leftarrow x^*$;
                16. $c_i \leftarrow c_i - 1$;
                17. else
                    18. $c_i \leftarrow c_i + 1$;
            else
                19. end
            end
         end
      end
   end
end

Complimentary Contributor Copy
Figure 6. Genetic Algorithm evolution (Killer Queen).

(see Fig. 7). Note that the Killer Queen algorithm reached the best score using the first parametric scheme. However, the difference in processing time is too large (around 10 times larger).

Figs. 8 to 10 show the cost function for each temperature of the simulated annealing with crystallization heuristic. Each graph shows three curves: the maximum, average and minimum cost function for each given temperature. All the three tests reach high scores.

Table 1 presents the initial and final cost values for 10 executions using the simulated annealing algorithm proposed and the parametric representation scheme proposed. It is possible to observe that, even when the initial cost is very low, the simulated annealing converges to a solution with very high score. Independently of the initial solution, higher scores are observed to be obtained when compared with the Killer Queen.
Figure 7. Genetic Algorithm evolution (Killer Queen).

Table 1. Score for the initial and final solution, for each of the 10 executions of the simulated annealing algorithm

<table>
<thead>
<tr>
<th>Case</th>
<th>Initial cost</th>
<th>Final cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>30.24</td>
<td>199.5</td>
</tr>
<tr>
<td>2</td>
<td>111.14</td>
<td>197.9</td>
</tr>
<tr>
<td>3</td>
<td>59.19</td>
<td>204.3</td>
</tr>
<tr>
<td>4</td>
<td>28.49</td>
<td>203.1</td>
</tr>
<tr>
<td>5</td>
<td>29.48</td>
<td>204.5</td>
</tr>
<tr>
<td>6</td>
<td>110.09</td>
<td>200.4</td>
</tr>
<tr>
<td>7</td>
<td>118.13</td>
<td>201.4</td>
</tr>
<tr>
<td>8</td>
<td>106.64</td>
<td>202.0</td>
</tr>
<tr>
<td>9</td>
<td>94.30</td>
<td>202.7</td>
</tr>
<tr>
<td>10</td>
<td>136.75</td>
<td>204.2</td>
</tr>
</tbody>
</table>
6. Best Model Construction and Test

A scale model of the best solution obtained by the simulated annealing was constructed. Fig. 11 shows the geometry of the best solution.

In this model, end plates were added and the fuselage was changed. The model was tested in a wind tunnel to verify if the coefficients obtained by the algorithm were correct. All the calculations are based on perfect conditions, distinct from reality, although the assumptions can be considered pessimistic. Fig. 12 shows the scale model tested.

In this test the forces and moments acting in the model were determined using an aerodynamic balance in an equivalent speed to that of the real aircraft in order to obtain the same value for the Reynolds number. They were tested in many angles for many flight configurations (with and without end plate and with more than one fuselage) in order to determine the best configuration.

The results were very satisfying, with a better performance than the one obtained using computational simulation, which was pessimistic in relation to the fuselage influence.
Figure 9. Test using the Simulated Annealing algorithm.

Figure 10. Test using Simulated Annealing test.
Figure 11. Best model obtained using simulated annealing.

Figure 12. Scale model testing in a wind tunnel.

7. Conclusion

The correct comparison to make is not between the genetic algorithm and simulated annealing, but between the two parametric representation schemes. The first one, used in the genetic algorithm, was incapable of consistently generating solutions that satisfied the restrictions. The parametric scheme proposed used by the simulated annealing satisfies every restriction along most of the process. The evolution in constraint optimizations is known to only occur if most of the solutions investigated are valid.

The combination between simulated annealing and the parametric representation scheme showed to be very effective, as convergence is reached in a short
period of time and the solutions are models with high score. The optimal project was validated through wind tunnel testing with a scale model.

References


Chapter 4

SIMULATED ANNEALING
APPLIED TO THE RESOLUTION
OF PHYLOGENETIC RECONSTRUCTION
WITH MAXIMUM PARSIMONY

Jean-Michel Richer, Karla E. Vazquez-Ortiz and David Lesaint*
LERIA, Université d’Angers, France

Abstract
In Bioinformatics, the Maximum Parsimony (MP) problem aims at reconstructing a phylogenetic tree from DNA sequences while minimizing the total number of genetic transformations. Different metaheuristics have been applied to solve this problem like tabu search, genetic and memetic algorithms. In this paper we propose a carefully devised simulated annealing implementation called SAMPARS (Simulated Annealing for Maximum PARSimony). Different possibilities for the key components (initial temperature, cooling schedule, improved neighborhood function), and input parameter values of SAMPARS were carefully analyzed and tuned. Its performance is investigated through extensive experimentation over well known benchmark instances showing that our algorithm is able to improve some previous best-known solutions.

*E-mail address: richer@info.univ-angers.fr, vazquez@info.univ-angers.fr, lesaint@info.univ-angers.fr

Complimentary Contributor Copy
1. Introduction

One of the main problems in Comparative Biology consists in establishing ancestral relationships between a group of \( n \) species or homologous genes in populations of different species, designated as taxa or OTU (Operational Taxonomic Units). These ancestral relationships are usually represented by a binary rooted tree, which is called a phylogenetic tree or a phylogeny [23].

In the past phylogenetic trees were inferred by using morphological characteristics like color, size, number of legs, etc. Nowadays, they are reconstructed using the information from biologic macromolecules like DNA (deoxyribonucleic acid), RNA (ribonucleic acid) and proteins. The problem of reconstructing molecular phylogenetic trees has become an important field of study in Bioinformatics and has many practical applications in population genetics, whole genome analysis, and the search for genetic predictors of disease [24, 41].

There exist many different methods reported in the literature, to solve the problem of reconstructing phylogenetic trees. These methods can be classified in three main different approaches. Distance methods [15, 36], Probabilistic methods [12, 38] and Cladistic methods [7, 11]. In this paper we focus our attention in a cladistic method based on the Maximum Parsimony (MP) criterion, which is considered in the literature as one of the most suitable evaluation criterion for phylogenies [28, 40]. The following definitions concern the Fitch parsimony, for which all changes have equal cost. Wagner or Camin-Sokal parsimony lead to different definitions. For a thorough introduction to the MP problem we refer the reader to [13].

Given a set \( \mathcal{S} = \{S_1, S_2, \ldots, S_n\} \) composed by \( n \) sequences of length \( k \) over a predefined alphabet \( \mathcal{A} \), a binary rooted phylogenetic tree \( T = (V,E) \) is used to represent their ancestral relationships, it consists of a set of nodes \( V = \{v_1, \ldots, v_r\} \) and a set of edges \( E \subseteq V \times V = \{\{u,v\} | u, v \in V\} \). The set of nodes \( V (|V| = (2n-1)) \) is partitioned into two subsets: \( I \), containing \( n-1 \) internal nodes or hypothetical ancestors, each having 2 descendants; and \( L \) which is composed of \( n \) taxa that are leaves of the tree, i.e., nodes with no descendant.

The parsimony sequence \( z = \{z_1, \ldots, z_k\} \) for each internal node \( z = f(x,y) \in I \), whose descendants are \( x = \{x_1, \ldots, x_k\} \) and \( y = \{y_1, \ldots, y_k\} \), is calculated...
with the following expression:

$$
\forall i, 1 \leq i \leq k, z_i = \begin{cases} 
  x_i \cup y_i, & \text{if } x_i \cap y_i = \emptyset \\
  x_i \cap y_i, & \text{otherwise}
\end{cases}.
$$

(1)

Then, the parsimony cost of the sequence \( z \) under Fitch optimality criterion is defined as follows:

$$
\phi(z) = \sum_{i=1}^{k} C_i 
$$

where

$$
C_i = \begin{cases} 
  1, & \text{if } x_i \cap y_i = \emptyset \\
  0, & \text{otherwise}
\end{cases}.
$$

(2)

and the parsimony cost for the tree \( T \) is obtained as follows:

$$
\phi(T) = \sum_{z \in \mathcal{I}} \phi(z).
$$

(3)

Thus, the Maximum Parsimony (MP) problem consists in finding a tree topology \( T^* \) for which \( \phi(T^*) \) is minimum, i.e.,

$$
\phi(T^*) = \min \{ \phi(T) : T \in \mathcal{T} \},
$$

(4)

where \( \mathcal{T} \) is the set composed by all the possible tree topologies also known as the search space of the problem.

Given a tree, the computation of the score of the tree, also known as the Small Parsimony Problem is a polynomial time algorithm. A rooted binary tree of \( n \) leaves has \( n - 1 \) internal nodes, thus the complexity of the small parsimony problem is \( O(n \times k) \).
In order to compute the overall cost (or score) of a tree (also known as tree length), Fitch’s algorithm [14] gradually moves back from the leaves to the root and computes hypothetical ancestral taxa $z \in I$. This is often referred to as the first-pass of the algorithm (see Fig. 1). The second-pass of the algorithm, which starts from the root and reaches the leaves assigns one nucleotide for a site if many possibilities exist, in order to obtain a hypothetical tree. However, only the first-pass is sufficient to obtain the parsimony score.

It has been demonstrated that the MP problem, also known as the Large Parsimony Problem, is NP-complete [21], since it is equivalent to the combinatorial optimization problem known as the Steiner tree problem on hypercubes, which is proven to be NP-complete [16].

The MP problem has been exactly solved for very small instances ($n \leq 10$) using a branch & bound algorithm (B&B) originally proposed by Hendy and Penny [22]. However, this algorithm becomes impractical when the number of studied species $n$ increases, since the size of the search space suffers a combinatorial explosion. Therefore, there is a need for heuristic methods to address the MP problem in reasonable time.

Andreatta and Ribeiro [4] compared three greedy algorithms of different complexity: 1stRotuGbr, Gstep_wR and Grstep. They concluded from their experiments that, Gstep_wR was more efficient than 1stRotuGbr, but expending more computational time. Grstep achieved good results only when it was combined with a local search method. Even when these methods attained good quality solutions, they were still far away from the optimal solutions.

In 2003, Barker proposed a software, called LVB, which implemented a multi-start simulated annealing algorithm for solving the MP problem [5]. Later, an updated version of LVB was released in 2010 [6]. This new version adds a hill-climbing phase at the end of each simulated annealing search and a new stop condition.

Ribeiro and Viana [30] in 2005 applied a greedy randomized adaptive search procedure (GRASP) for solving the MP problem and showed that this algorithm had the best performance with respect to the state-of-the-art algorithms. Different evolutionary algorithms were also reported for the MP problem. Among them we found GA+PR+LS, a genetic algorithm hybridized with local search which employs path-relinking to implement a progressive crossover operator [31]. More recently Richer, Goëffon and Hao [32] introduced a memetic al-
algorithm called Hydra which yields the best-known solutions for a set of 20 benchmark instances proposed in [29].

This paper describes a new simulated annealing (SA) algorithm implementation (hereafter called SAMPARS) for finding near-optimal solutions for the MP problem under the Fitch’s criterion. To achieve this, different possibilities for its key components were carefully designed and evaluated. The SAMPARS input parameter values yielding the best quality solutions to the problem at a reasonable computational effort were determined by employing a tuning methodology based on Combinatorial Interaction Testing. The performance of the new proposed implementation is investigated through extensive experimentation over 20 well known benchmark instances and compared with other existing state-of-the-art algorithms, showing that our algorithm is able to improve some previous best-known solutions.

The rest of this paper is organized as follows. In section 2, the components of our SAMPARS implementation are discussed in detail. Then, three computational experiments are presented in section 3, devoted to determine the best parameter settings for SAMPARS and to compare its performance with respect to LVB, an existing SA implementation [5,6] and two other representative state-of-the-art algorithms: GA+PR+LS [31] and Hydra [18]. Finally, the last section summarizes the main contributions of this work and presents some possible directions for future research.

2. An Improved Implementation of a Simulated Annealing Algorithm

Simulated Annealing (SA) is a general-purpose stochastic optimization technique that has proved to be an effective tool for the approximation of global optimal solutions to many NP-hard optimization problems. However, it is well known that developing an effective SA algorithm requires a careful implementation of some essential components and an appropriate tuning of the parameters used [25,26].

In this section we present an improved implementation of a SA algorithm (see Algorithm 1), that we called SAMPARS, to find tree topologies (phylogenies) with near-optimal parsimony costs under the Fitch optimality criterion. The main difference of our implementation, with respect to LVB [5,6], occurs in the neighborhood function (line 9) which has been tailored to fit the specificity
of the MP problem. SAMPARS employs a composed neighborhood function combining standard neighborhood relations for trees with a stochastic descent algorithm on the current solution (see section 2.4.), while LVB randomly selects a neighbor \( s' \in \mathcal{F} \) of the current solution \( s \).

**Algorithm 1: SAMPARS algorithm**

```plaintext
input: \( N \): neighborhood, \( f \): fitness function, \( \alpha \): cooling scheme, \( T_i \): initial temperature, \( T_f \): final temperature
1 \( s_0 \leftarrow \text{GenerateInitialSolution}() \)
2 \( s \leftarrow s_0 // \text{current solution} \)
3 \( s^* \leftarrow s_0 // \text{best solution} \)
4 \( t \leftarrow T_i // \text{set initial temperature} \)
5 repeat
6   \( c \leftarrow 0 \)
7   while \( c < CL \) do
8     \( c \leftarrow c + 1 \)
9     \( s' \leftarrow \text{GenerateNeighbor}(s, c, N) \)
10    \( \Delta f \leftarrow f(s') - f(s) \)
11    generate a random \( u \in [0, 1] \)
12    if \( \Delta f < 0 \) or \( (e^{-\Delta f / t} > u) \) then
13      \( s \leftarrow s' \)
14      if \( f(s') < f(s^*) \) then \( s^* \leftarrow s' \)
15   end
16 end
17 \( t \leftarrow \alpha t \)
18 until \( t > T_f \)
19 return \( s^* \)
```

In the remainder of this section all the implementation details of the SAMPARS algorithm are presented. For some of these components different possibilities were analyzed (see section 3.2.) in order to find the combination of them which offers the best quality solutions at a reasonable computational effort.

### 2.1. Internal Representation and Search Space

Let \( T \) be a potential solution in the search space \( \mathcal{T} \), that is a phylogenetic tree representing the ancestral relationships of a group of \( n \) operational taxa, each one of length \( k \) over a predefined alphabet \( \mathcal{A} \). Then \( T \) is represented as a binary rooted tree composed of \( n - 1 \) internal nodes and \( n \) leaves. The size of the search
space $|\mathcal{T}|$, i.e., the number of rooted tree topologies is given by the following expression [46]:

$$|\mathcal{T}| = \frac{(2n - 3)!}{2^{n-2}(n-2)!}$$

(5)

### 2.2. Evaluation Function

The evaluation function is one of the key elements for the successful implementation of metaheuristic algorithms because it is in charge of guiding the search process toward good solutions in a combinatorial search space.

Previous works for solving the MP problem have commonly evaluated the quality of a potential solution, $\phi(T)$, using the parsimony cost depicted in (3) [4, 5, 30–32]. In our SAMP ARS implementation this evaluation function was also used.

### 2.3. Initial Solution

The initial solution is the starting phylogenetic tree used for the algorithm to begin the search for better configurations in the search space $\mathcal{T}$.

In the existing SA implementation for the MP problem [5, 6] the initial solution is randomly generated. In contrast, SAMP ARS creates the starting solution using a greedy procedure that guarantees a better quality of the initial solution. The proposed procedure can be described as follows.

First, we generate a random permutation of the studied taxa, which is used to indicate the order in which the leaves (taxa) will be selected. Then, the root node of the tree is created and the first and second taxa in the permutation are bound to it. The rest of the taxa in the permutation are added to the tree one by one. Each time a new taxon is added to the partial tree, the algorithm analyzes all the possible insertion positions in order to select the one that minimizes the increase in the tree’s parsimony cost. This process is iterated until all the remaining taxa in the permutation are processed.

Other initialization methods can be used to generate a starting configuration

- **upgma, nj**: distance methods like UPGMA [39] or Neighbor-Joining (NJ) [17, 37] as they have a low complexity
• \textit{nj + fitch}: use of distance method NJ where the distance is the fitch score instead of the hamming distance: at each step of the NJ algorithm we recompute the distance matrix using fitch scoring function

• \textit{nj + greedy}: we have also designed a Neighbor-Joining and greedy procedure which uses half of the sequences to generate a first NJ tree, the second half is added to the tree obtained by NJ in a greedy manner

2.4. Neighborhood Functions

The most common practice in the reported metaheuristics for the MP problem [4, 30, 31] is to employ one of the following three neighborhood functions (see Fig. 2). The first one, called Nearest Neighbor Interchange (NNI), was proposed by Waterman and Smith [45]. It exchanges two subtrees separated by an internal node. Given that each tree has \( n - 3 \) internal nodes and two possible moves by branch, then there exist \( 2n - 6 \) NNI neighboring solutions [33]. The second one, is known as Subtree Pruning and Regrafting (SPR) [42]. It cuts a branch of the tree and reinserts the resulting subtree elsewhere generating a new internal node. For each tree there exist \( 2(n - 3)(2n - 7) \) possible SPR neighboring solutions [3]. Finally, the Tree Bisection and Reconnection (TBR) [42] consists in dividing the tree into two subtrees that will be reconnected from one of their branches. From a given tree, the TBR neighborhood induces at most \( (2n - 3)(n - 3)^2 \) neighboring trees [3].

LVB, an existing SA implementation for the MP problem [5, 6] alternates the use of the NNI and SPR neighborhood functions at each iteration of the search process. In the case of our SAMPARS algorithm both the SPR and the TBR neighborhood relations are implemented. However, from our preliminary experiments it has been observed that the separated use of these neighborhood functions is not sufficient to reach the best-known solutions, because both of them are highly disruptive. In order to achieve a better performance for SAMPARS, we have decided to use a third complementary neighborhood structure. It is based on a stochastic descent algorithm with a best-improve scheme (see Algorithm 2) which is occasionally applied to the neighboring solution \( s' \) prior to returning it. Our neighborhood function is inspired by the ideas reported in [27], where the advantage of using this approach is well documented.
Algorithm 2: GenerateNeighbor

input: s: current solution, c: number of neighbors visited, \( N \): neighborhood

1. randomly select \( s' \in N(T) \) // get a SPR or TBR neighboring solution
2. if \( c \) is a multiple of 25 then
   3. \( s' \leftarrow \text{Descent}(s') \)
3. end
4. return \( s' \)
2.5. Cooling Schedule

A cooling schedule is defined by the following parameters: an initial temperature $T_i$, a final temperature $T_f$ or a stopping criterion, the maximum number of neighboring solutions that can be generated at each temperature (Markov chain length), and a rule for decrementing the temperature. The cooling schedule governs the convergence of the SA algorithm. At the beginning of the search, when the temperature is high, the probability of accepting solutions of worse quality than the current solution (uphill moves) is high. It allows the algorithm to escape from local minima. The probability to accept such moves is gradually decreased as the temperature decreases to zero.

The literature offers a number of different cooling schedules, see for instance [1, 2, 34, 43]. They can be divided into two main categories: static and dynamic. In a static cooling schedule, the parameters are fixed and cannot be changed during the execution of the algorithm. With a dynamic cooling schedule the parameters are adaptively changed during the execution.

In the SAMPARS implementation we preferred a geometrical cooling scheme (static) mainly for its simplicity. It starts at an initial temperature $T_i$ that can either be defined by the user or automatically computed using the following formula: $(k + n)^{1.0/3.3}$ which generates values under 6.0 for most of the tested benchmark instances. Then, this temperature is decremented at each round by a factor $\alpha = 0.99$ using the relation $t = \alpha t$. A reheat mechanism has also been implemented. If the best-so-far solution is not improved during $\text{maxNITD} = 50$ consecutive temperature decrements, the current temperature $t_j$ is increased by a factor $\beta = 1.4$ using the function $t = \beta t$. In our implementation this reheat mechanism can be applied at most $\text{maxReheat} = 3$ times, since it represents a good trade-off between efficiency and quality of the solutions found.

For each temperature $t_j$, the maximum number of visited neighboring solutions is $CL$. It depends directly on the parameters $n$ and $k$ of the studied instance, since we have observed that more moves are required for bigger trees [44]. The three different values that $CL$ can take, and that were empirically decided, are:

- small: $CL = 15(n + k)$
- medium: $CL = 23(n + k)$
- large: $CL = 40(n + k)$
The parameter values presented in this section were chosen based on the results obtained in a preliminary experimentation. For the reason of space limitation we did not present here these experiments.

2.6. Stop Condition

The SAMPARS algorithm stops if it ceases to make progress. In our implementation a lack of progress exists if after $\omega = 40$ (frozen factor) consecutive temperature decrements the best-so-far solution is not improved.

We will see later that thanks to the main features presented in this section, the SAMPARS algorithm reaches good quality results, which are sometimes better than the best-known solutions reported in the literature [18, 31].

3. Computational Experiments

In this section three main experiments were conducted to evaluate the performance of the proposed SAMPARS algorithm and some of its components are presented. The objective of the first experiment is to determine both a component combination, and a set of parameter values which enables SAMPARS to attain the best trade-off between solution quality and computational effort. The purpose of the second experiment is to carry out a performance comparison of SAMPARS with respect to an existing SA algorithm called LVB [5, 6]. The third experiment is devoted to asses the performance of SAMPARS with respect to two representative state-of-the-art procedures: Hydra [18] and GA+PR+LS [31].

For these experiments SAMPARS was coded in C++ and compiled with g++ using the optimization flag -O3. It was run sequentially into a CPU Xeon X5650 at 2.66 GHz, 2 GB of RAM with Linux operating system. Due to the non-deterministic nature of the studied algorithms, 30 independent runs were executed for each of the selected benchmark instances in each experiment presented in this section.

3.1. Benchmark Instances and Performance Assessment

The test-suites that we have used in our experiments are the same proposed by Ribeiro and Vianna [29, 30] and later employed in other works [18, 31]. It
consists of two sets. A first set (set 1) of 20 randomly generated instances (tst01 to tst20) with a number of sequences (n) ranging from 45 to 75 whose length (k) varies from 61 to 159. A second set (set 2) of 8 benchmarks obtained from real data.

For all the experiments, 30 independent executions were performed. The criteria used for evaluating the performance of the algorithms are the same as those used in the literature: the best parsimony cost found for each instance (smaller values are better) and the expended CPU time in seconds.

3.2. Components and Parameters Tuning

Optimizing parameter settings is an important task in the context of algorithm design. Different procedures have been proposed in the literature to find the most suitable combination of parameter values [10,20]. In this paper we employ a tuning methodology based on Combinatorial Interaction Testing (CIT) [8]. We have decided to use CIT, because it allows to significantly reduce the number of tests (experiments) needed to determine the best parameter settings of an algorithm. Instead of exhaustively testing all the parameter value combinations of the algorithm, it only analyzes the interactions of t (or fewer) input parameters by creating interaction test-suites that include at least once all the t-way combinations between these parameters and their values.

Covering arrays (CAs) are combinatorial designs which are extensively used to represent those interaction test-suites. A covering array, CA(N; t, k, v), of size N, strength t, degree k, and order v is an N × k array on v symbols such that every N × t sub-array includes, at least once, all the ordered subsets from v symbols of size t (t-tuples) [9]. The minimum N for which a CA(N; t, k, v) exists is the covering array number and it is defined according to the following expression: CAN(t, k, v) = min{N : ∃CA(N; t, k, v)}.

CAs are used to represent an interaction test-suite as follows. In an algorithm we have k input parameters. Each of these has v values or levels. An interaction test-suite is an N × k array where each row is a test case. Each column represents an input parameter and a value in the column is the particular configuration. This test-suite allows to cover all the t-way combinations of input parameter values at least once. Thus, the costs of tuning the algorithm can be substantially reduced by minimizing the number of test cases N in the covering array.
Table 1. Input parameters of the SAMPARS algorithm and their selected values.

<table>
<thead>
<tr>
<th>IS</th>
<th>$\alpha$</th>
<th>$i_t$</th>
<th>$N$</th>
<th>CL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greedy</td>
<td>0.99</td>
<td>6.0</td>
<td>SPR</td>
<td>$15(n+k)$</td>
</tr>
<tr>
<td>Random</td>
<td>0.85</td>
<td>$(k+n)^{1/3}$</td>
<td>TBR</td>
<td>$23(n+k)$</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>$40(n+k)$</td>
</tr>
</tbody>
</table>

In practice, algorithms’ input parameters do not have exactly the same number of values (levels). To overcome this limitation of CAs, mixed level covering arrays (MCAs) are used. A MCA($N; t, k, (v_1, v_2, \ldots, v_k)$) is an $N \times k$ array on $v$ symbols ($v = \sum_{i=1}^{k} v_i$), where each column $i$ ($1 \leq i \leq k$) of this array contains only elements from a set $S_i$, with $|S_i| = v_i$. This array has the property that the rows of each $N \times t$ sub-array cover all $t$-tuples of values from the $t$ columns at least once. Next, we present the details of the tuning process, based on CIT, for the particular case of our SAMPARS algorithm.

First, we have identified $k = 5$ input parameters used for SAMPARS: initial solution procedure $IS$, cooling factor $\alpha$, initial temperature $T_i$, neighborhood function $N$, and maximum number of visited neighboring solutions $CL$. Based on some preliminary experiments, certain reasonable values were selected for each one of those input parameters (shown in Table 1).

The smallest possible mixed level covering array MCA($24; 4, 5, (2, 2, 2, 2, 3)$), shown (transposed) in Table 2, was obtained by using the Memetic Algorithm reported in [35]. This covering array can be easily mapped into an interaction test-suite by replacing each symbol from each column to its corresponding parameter value. For instance, we can map 0 in the first column (the first line in Table 2) to Greedy and 1 to Random. The resulting interaction test-suite contains, thus, 24 test cases (parameter settings) which include at least once all the 4-way combinations between SAMPARS’s input parameters and their values$^1$.

Each one of those 24 test cases was used to executed 30 times the SAMPARS algorithm over the 20 instances of the test-suite described in section 3.1.

---

$^1$In contrast, with an exhaustive testing which contains $3(2^4) = 48$ test cases.
Table 2. Mixed level covering array $MCA(24; 4, 5, (2, 2, 2, 2, 3))$ representing an interaction test-suite for tuning SAMPARS (transposed).

<table>
<thead>
<tr>
<th>1 0 1 1 1 0 0 0 1 0 1 0 1 0 1 0 0 0 0 1 0 1 1</th>
<th>1 0 0 1 0 0 1 0 0 1 0 1 0 1 0 1 0 1 0 1 1 1 1 1</th>
<th>1 1 0 1 0 1 1 1 0 1 1 0 0 0 0 1 1 1 0 0 0</th>
<th>0 0 0 1 1 1 0 0 1 1 1 0 1 0 0 1 1 0 0 1 0 1 0 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 1 0 1 1 1 0 0 2 1 0 0 2 2 2 0 2 1 1 1 2 0 0 2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 3. Average results obtained in the tuning experiments using 24 test cases over 20 standard benchmark instances.

The data generated by these 14400 executions is summarized in Fig. 3, which depicts the average cost reached over the selected instances by each test case.

From this graphic we have selected the 5 test cases which yield the best results. Their average parsimony cost and the average CPU time in seconds are presented in Table 3. This table allowed us to observe that the parameter setting giving the best trade-off between solution quality and computational effort corresponds to the test case number 17 (shown in bold). The best average parsimony cost with an acceptable speed is thus reached with the following input parameter values: initial solution procedure $IS = Greedy$, cooling factor $\alpha = 0.99$, initial temperature $T_i = 6.0$, neighborhood function $N = TBR$ and maximum
Simulated Annealing Applied to the Resolution of Phylogenetic ...

Table 3. Results from the 5 best parameter test cases in the tuning experiment.

<table>
<thead>
<tr>
<th>Num.</th>
<th>Test case</th>
<th>Avg. parsimony</th>
<th>Avg. CPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>00012</td>
<td>1004.06</td>
<td>3512.51</td>
</tr>
<tr>
<td>14</td>
<td>10002</td>
<td>1004.14</td>
<td>3511.35</td>
</tr>
<tr>
<td>10</td>
<td>00011</td>
<td>1005.00</td>
<td>2058.93</td>
</tr>
<tr>
<td>3</td>
<td>10001</td>
<td>1005.02</td>
<td>2047.52</td>
</tr>
<tr>
<td>9</td>
<td>10112</td>
<td>1005.29</td>
<td>3136.40</td>
</tr>
</tbody>
</table>

number of visited neighboring solutions $CL = 40(n+k)$. These values are thus used in the experimentation reported next.

3.3. Comparing SAMPARS with an Existing SA Implementation

For this experiment a subset of six representative benchmark instances, taken from the test-suite described in Sect. 3.1., was selected (comparable results were obtained with all the other tested instances). Then, the latest version of LVB was obtained, compiled and executed in our computational platform using the input parameters suggested by their authors [6]. SAMPARS was also executed over the same six instances.

Table 4 displays the detailed computational results produced by this experiment. The first three columns in the table indicate the name of the instance as well as its number of taxa ($n$) and length ($k$). For each compared algorithm the best ($B$), average ($Avg.$), and standard deviation ($Dev.$) of the parsimony cost attained in 30 independent executions and its average CPU time in seconds are listed in columns 4 to 11. A statistical significance analysis was performed for this experiment. First, D’Agostino-Pearson’s omnibus $K^2$ test was used to evaluate the normality of data distributions. For normally distributed data, either ANOVA or the Welch’s $t$ parametric tests were used depending on whether the variances across the samples were homogeneous ($homoskedasticity$) or not. This was investigated using the Bartlett’s test. For non-normal data, the non-parametric Kruskal-Wallis test was adopted. A significance level of 0.05 has been considered. The resulting $P$-value is presented in Column 12. Last column shows a “+” symbol if there exists a statistically significant increase in
performance achieved by SAMPARS with regard to LVB, the existing SA Implementation.

Table 4. Comparison between SAMPARS and LVB (an existing SA implementation [5,6]) over a subset of six selected instances

<table>
<thead>
<tr>
<th>Instance</th>
<th>n</th>
<th>k</th>
<th>B</th>
<th>Avg.</th>
<th>Dev.</th>
<th>T</th>
<th>B</th>
<th>Avg.</th>
<th>Dev.</th>
<th>T</th>
<th>P-value</th>
<th>SS</th>
</tr>
</thead>
<tbody>
<tr>
<td>tst01</td>
<td>45</td>
<td>61</td>
<td>549</td>
<td>553.87</td>
<td>2.47</td>
<td>85.57</td>
<td>545</td>
<td>545.83</td>
<td>0.87</td>
<td>295.80</td>
<td>1.80E-11</td>
<td>+</td>
</tr>
<tr>
<td>tst02</td>
<td>47</td>
<td>151</td>
<td>1367</td>
<td>1375.33</td>
<td>4.77</td>
<td>23.63</td>
<td>1354</td>
<td>1356.13</td>
<td>1.33</td>
<td>479.50</td>
<td>5.08E-21</td>
<td>+</td>
</tr>
<tr>
<td>tst03</td>
<td>49</td>
<td>111</td>
<td>840</td>
<td>850.83</td>
<td>4.86</td>
<td>68.02</td>
<td>833</td>
<td>834.00</td>
<td>1.05</td>
<td>577.12</td>
<td>1.38E-18</td>
<td>+</td>
</tr>
<tr>
<td>tst08</td>
<td>57</td>
<td>119</td>
<td>867</td>
<td>879.80</td>
<td>5.01</td>
<td>922.61</td>
<td>852</td>
<td>854.53</td>
<td>2.37</td>
<td>665.50</td>
<td>2.12E-11</td>
<td>+</td>
</tr>
<tr>
<td>tst09</td>
<td>59</td>
<td>93</td>
<td>1153</td>
<td>1160.77</td>
<td>4.60</td>
<td>58.53</td>
<td>1143</td>
<td>1145.50</td>
<td>1.11</td>
<td>719.24</td>
<td>3.48E-18</td>
<td>+</td>
</tr>
<tr>
<td>tst10</td>
<td>60</td>
<td>71</td>
<td>727</td>
<td>738.00</td>
<td>5.59</td>
<td>570.62</td>
<td>720</td>
<td>721.27</td>
<td>0.78</td>
<td>500.28</td>
<td>1.89E-16</td>
<td>+</td>
</tr>
</tbody>
</table>

From Table 4 we can observe that SAMPARS is the most time-consuming algorithm, since it uses an average of 539.57 seconds for solving these six instances. On the contrary, LVB employs only 288.16 seconds. However, we can also remark that SAMPARS can take advantage of its longer executions. Indeed it is able to consistently improve the best results found by LVB, obtaining in certain instances, like tst08, an important decrease in the parsimony cost (up to $-15 = 852 - 867$). Furthermore, the solution cost found by SAMPARS presents a relatively small standard deviation (see Column Dev.). It is an indicator of the algorithm’s precision and robustness since it shows that in average the performance of SAMPARS does not present important fluctuations.

The statistical analysis presented in the last two columns of Table 4 confirms that there exists a statistically significant increase in performance achieved by SAMPARS with regard to LVB on the six studied instances. Thus, we can conclude that SAMPARS is more effective than the existing SA algorithm reported in [5,6]. Below, we will present more computational results obtained from a performance comparison carried out between SAMPARS and some state-of-the-art procedures.
3.4. Comparing SAMPARS with the State-of-the-art Procedures

In this experiment a performance comparison of the best solutions achieved by SAMPARS with respect to those produced by GA+PR+LS [31] and Hydra [18] was carried out over the test-suite described in Sect. 3.1. The results from this experiment are depicted in Table 5. Columns 1 to 3 indicate the instance and its size in terms of taxa \((n)\) and length \((k)\). The best solutions found by GA+PR+LS and Hydra, in terms of parsimony cost \(\Phi\) are listed in the next two columns. Columns 6 to 9 present the best \((B)\), average \((\text{Avg.})\), and standard deviation \((\text{Dev.})\) of the parsimony cost attained by SAMPARS in 30 independent executions, as well as its average CPU time in seconds. Finally, the difference \((\delta)\) between the best result produced by our SAMPARS algorithm and the best-known solution produced by either GA+PR+LS or Hydra is shown in the last column.

Table 5. Performance comparison among SAMPARS, GA+PR+LS [31] and Hydra [18] over 20 standard benchmark instances

<table>
<thead>
<tr>
<th>Instance</th>
<th>(n)</th>
<th>(k)</th>
<th>GA+PR+LS</th>
<th>TNT</th>
<th>Hydra</th>
<th>SAMPARS</th>
<th>B</th>
<th>Avg.</th>
<th>Dev.</th>
<th>(T)</th>
<th>(\delta)</th>
</tr>
</thead>
<tbody>
<tr>
<td>tst01</td>
<td>45</td>
<td>61</td>
<td>547</td>
<td>1000</td>
<td>545</td>
<td>545</td>
<td>545</td>
<td>545.13</td>
<td>0.43</td>
<td>1407.57</td>
<td>0</td>
</tr>
<tr>
<td>tst02</td>
<td>47</td>
<td>151</td>
<td>1361</td>
<td>1354</td>
<td>1354</td>
<td>1354</td>
<td>1355.30</td>
<td>0.97</td>
<td>1938.23</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>tst03</td>
<td>49</td>
<td>111</td>
<td>837</td>
<td>833</td>
<td>833</td>
<td>833</td>
<td>833.43</td>
<td>0.56</td>
<td>2506.30</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>tst04</td>
<td>50</td>
<td>97</td>
<td>590</td>
<td>588</td>
<td>588</td>
<td>587</td>
<td>588.23</td>
<td>0.80</td>
<td>1341.17</td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>tst05</td>
<td>52</td>
<td>75</td>
<td>792</td>
<td>789</td>
<td>789</td>
<td>789</td>
<td>789.00</td>
<td>0.00</td>
<td>2007.90</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>tst06</td>
<td>54</td>
<td>65</td>
<td>603</td>
<td>596</td>
<td>596</td>
<td>596</td>
<td>596.57</td>
<td>0.56</td>
<td>1164.27</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>tst07</td>
<td>56</td>
<td>143</td>
<td>1274</td>
<td>1269</td>
<td>1269</td>
<td>1269</td>
<td>1270.83</td>
<td>1.63</td>
<td>4063.80</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>tst08</td>
<td>57</td>
<td>119</td>
<td>862</td>
<td>852</td>
<td>852</td>
<td>852</td>
<td>853.33</td>
<td>1.27</td>
<td>2884.73</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>tst09</td>
<td>59</td>
<td>93</td>
<td>1150</td>
<td>1144</td>
<td>1144</td>
<td>1144</td>
<td>1144.73</td>
<td>1.09</td>
<td>3237.53</td>
<td>-3</td>
<td></td>
</tr>
<tr>
<td>tst10</td>
<td>60</td>
<td>71</td>
<td>722</td>
<td>721</td>
<td>721</td>
<td>720</td>
<td>720.80</td>
<td>0.70</td>
<td>2288.00</td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>tst11</td>
<td>62</td>
<td>63</td>
<td>547</td>
<td>542</td>
<td>542</td>
<td>541</td>
<td>542.21</td>
<td>0.72</td>
<td>3807.79</td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>tst12</td>
<td>64</td>
<td>147</td>
<td>1225</td>
<td>1211</td>
<td>1211</td>
<td>1208</td>
<td>1215.27</td>
<td>2.76</td>
<td>3668.40</td>
<td>-3</td>
<td></td>
</tr>
<tr>
<td>tst13</td>
<td>65</td>
<td>113</td>
<td>1524</td>
<td>1515</td>
<td>1515</td>
<td>1515</td>
<td>1517.77</td>
<td>1.91</td>
<td>2514.20</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>tst14</td>
<td>67</td>
<td>99</td>
<td>1171</td>
<td>1160</td>
<td>1160</td>
<td>1160</td>
<td>1163.03</td>
<td>1.82</td>
<td>2847.13</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>tst15</td>
<td>69</td>
<td>77</td>
<td>758</td>
<td>752</td>
<td>752</td>
<td>752</td>
<td>753.90</td>
<td>1.11</td>
<td>4808.63</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>tst16</td>
<td>70</td>
<td>69</td>
<td>537</td>
<td>529</td>
<td>529</td>
<td>529</td>
<td>531.00</td>
<td>1.23</td>
<td>3268.20</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>tst17</td>
<td>71</td>
<td>159</td>
<td>2469</td>
<td>2453</td>
<td>2453</td>
<td>2450</td>
<td>2456.00</td>
<td>2.63</td>
<td>8020.23</td>
<td>-3</td>
<td></td>
</tr>
<tr>
<td>tst18</td>
<td>73</td>
<td>117</td>
<td>1531</td>
<td>1522</td>
<td>1522</td>
<td>1521</td>
<td>1525.67</td>
<td>3.96</td>
<td>4451.37</td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>tst19</td>
<td>74</td>
<td>95</td>
<td>1024</td>
<td>1013</td>
<td>1013</td>
<td>1012</td>
<td>1016.23</td>
<td>2.14</td>
<td>6875.30</td>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>tst20</td>
<td>75</td>
<td>79</td>
<td>671</td>
<td>661</td>
<td>661</td>
<td>659</td>
<td>662.82</td>
<td>1.44</td>
<td>7149.43</td>
<td>-2</td>
<td></td>
</tr>
</tbody>
</table>

Avg. 1009.75 1002.45 1001.65 1004.06 1.39 3512.51

Complimentary Contributor Copy
The analysis of the data presented in Table 5 lead us to the following observations. First, we clearly remark that the procedure GA+PR+LS [31] consistently returns poorer quality solutions than Hydra and SAMPARS. Second, the best solution quality attained by the proposed SAMPARS algorithm is very competitive with respect to that produced by the existing state-of-the-art procedure called Hydra [18], since in average SAMPARS provides solutions whose parsimony costs are smaller (compare Columns 5 and 6). In fact, it is able to improve on 9 previous best-known solutions produced by Hydra and to equal these results for the other 11 benchmark instances.

Thus, as this experiment confirms, our SAMPARS algorithm is an effective alternative for solving the MP problem, compared with the two more representative state-of-the-art algorithms: GA+PR+LS [31] and Hydra [18].

3.5. Delving Deeper into SAMPARS

3.5.1. Influence of the Frequency of the Best Improvement

In section 2.4. we have explained the influence of the GenerateNeighbor function based on a descent algorithm with a best improve scheme applied with some frequency. For example, every 25 iterations we improve the new configuration with a descent algorithm. This mechanism seems to greatly improve the results obtained. We have decided to study the influence of the frequency of improvements and make it vary from 5, 10-100, 200-600.

Results obtained for different instances (see Table 6 and 7 for tst12) show that:

- the computation time increases with a small frequency \( f \) as there are more descents to perform: for example for instance tst12, with a frequency of 5, the computation time is around 72 minutes, while with a frequency of 10 it is only of 31 minutes,

- if the frequency is too low (\( f < 20 \)) then it is more difficult to reach the best known optimal solution as we get stuck in local optima. For \( f = 10 \), the average score of the solutions is 1226.50 and the minimum score found is 1222 while for \( f = 20 \) the average score is 1219 and the minimum score found reaches 1213,
Simulated Annealing Applied to the Resolution of Phylogenetic...

Table 6. Results obtained for tst12 with nj+greedy for $f_{requency} = 5,\ldots,70$

<table>
<thead>
<tr>
<th>freq.</th>
<th>5</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
</tr>
</thead>
<tbody>
<tr>
<td>time (m)</td>
<td>72</td>
<td>31</td>
<td>17</td>
<td>13</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>average</td>
<td>1227.70</td>
<td>1226.50</td>
<td>1219.90</td>
<td>1218.70</td>
<td>1218.50</td>
<td>1216.10</td>
<td>1214</td>
<td>1214</td>
</tr>
<tr>
<td>minimum</td>
<td>1222</td>
<td>1222</td>
<td>1215</td>
<td>1215</td>
<td>1213</td>
<td>1213</td>
<td>1210</td>
<td>1208</td>
</tr>
</tbody>
</table>

Table 7. Results obtained for tst12 with nj+greedy for $f_{requency} = 80,\ldots,600$

<table>
<thead>
<tr>
<th>freq.</th>
<th>80</th>
<th>90</th>
<th>100</th>
<th>200</th>
<th>300</th>
<th>400</th>
<th>500</th>
<th>600</th>
</tr>
</thead>
<tbody>
<tr>
<td>time (m)</td>
<td>8</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>average</td>
<td>1217.40</td>
<td>1218.30</td>
<td>1218.40</td>
<td>1219.80</td>
<td>1217.40</td>
<td>1220.30</td>
<td>1220.10</td>
<td>1220.10</td>
</tr>
<tr>
<td>minimum</td>
<td>1215</td>
<td>1216</td>
<td>1215</td>
<td>1212</td>
<td>1216</td>
<td>1213</td>
<td>1213</td>
<td>1215</td>
</tr>
</tbody>
</table>

- if the frequency is too high ($f > 200$) then it is more difficult to reach a good solution,
- depending on the problem, the interval for which we can obtain good results is $20 \leq f \leq 100$.

3.5.2. Influence of the Reheat Mechanism and effort

In order to determine the influence of the reheat mechanism we run a total of 26,000 tests for medium and large efforts, different initialization methods ($random, nj, greedy, nj + greedy, nj + fitch$), with different improvement frequencies ($20, 30, 60, 90$) and reheat values that range from 0 (no reheat) to 4 (reheat is performed at most 4 times) and a SPR neighborhood. Results are shown on Table 8. For each problem we provide the average parsimony score obtained, the minimum score that could be reached and the number of times it was reached.

We can see that tst10 is not a complex problem because we could find the minimum score of 720 for every case. However, we can see that a reheat of 4

Complimentary Contributor Copy
enabled to find the minimum score 38 times over all tests performed, i.e. a total of 400 (5 methods, 4 frequencies and 20 executions). Problem tst20 is a more complex problem as the minimum score of 659 could only be found with a large effort with a reheat of 1 or 4.

The results confirm our former observations: a large effort will lead to results of better quality compared to a medium effort and a reheat of 1 or 4 can provide better results.

3.5.3. Influence of initial solution

As a first step, we have tried to determine which initialization procedure was the most interesting. The initial solution can sometimes influence the search. For example, with a descent algorithm (local search) that employs a NNI (Nearest Neighbor Interchange) neighborhood, if one starts from a solution with a high parsimony score then the final solution will generally be of poor quality as the algorithm will converge prematurely to a local optimum [19].

Tests performed on set 2 (see Table 9) show that on average random > upgma > nj > nj + greedy > nj + fitch > greedy. In other words, the random initialization procedure produces initial solution of worst quality while a greedy algorithm gives the best configurations. For set 1 (results not reported here), we have slightly different results: random > upgma > nj > greedy > nj > greedy > nj + fitch.

In a second step, we have tried to answer the following question: does the initial solution have some influence on the final result? The answer, obtained from section 3.5.2, is that the nj and nj + greedy methods will on average provide better results and help reach a maximum number of good solutions. On Table 10 for each initialization method, effort and reheat we have provided the number of times the method gave the best results and the total of best results for problems tst10 to tst20.

4. Conclusions

In this paper we have presented an improved simulated annealing algorithm called SAMPARS to find near-optimal solutions for the MP problem under the Fitch optimality criterion. SAMPARS’s components and parameter values were
Table 8. Performance comparison for the reheat mechanism for problems tst10 to tst20

<table>
<thead>
<tr>
<th>effort</th>
<th>medium</th>
<th>large</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0 1 2 3 4</td>
<td>0 1 2 3 4</td>
</tr>
<tr>
<td>reheat</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tst10</td>
<td>723.12 722.67 722.70 722.81 722.52</td>
<td>722.33 722.10 722.05 721.98 721.99</td>
</tr>
<tr>
<td></td>
<td>12 23 14 10 16</td>
<td>23 27 36 32 38</td>
</tr>
<tr>
<td>tst11</td>
<td>544.73 544.20 544.09 543.98 544.00</td>
<td>543.96 543.62 543.58 543.41 543.42</td>
</tr>
<tr>
<td></td>
<td>5 5 4 5 1</td>
<td>11 2 6 2 1</td>
</tr>
<tr>
<td>tst12</td>
<td>1219.29 1218.91 1218.86 1218.59 1218.57</td>
<td>1217.37 1216.91 1217.21 1216.95 1216.92</td>
</tr>
<tr>
<td></td>
<td>1 1 2 1 1</td>
<td>1 4 3 3 2</td>
</tr>
<tr>
<td>tst13</td>
<td>1522.27 1521.94 1522.00 1521.88 1521.11</td>
<td>1520.38 1520.47 1520.50 1520.47</td>
</tr>
<tr>
<td></td>
<td>2 4 6 5 4</td>
<td>8 17 11 12 12</td>
</tr>
<tr>
<td>tst14</td>
<td>1167.48 1166.90 1166.53 1166.74 1166.58</td>
<td>1166.15 1165.58 1165.48 1165.37 1165.33</td>
</tr>
<tr>
<td></td>
<td>2 2 2 1 1</td>
<td>4 2 4 4 5</td>
</tr>
<tr>
<td>tst15</td>
<td>757.40 757.03 756.95 757.04 756.90</td>
<td>756.12 755.99 755.81 755.62 755.75</td>
</tr>
<tr>
<td></td>
<td>6 8 6 6 6</td>
<td>6 8 13 16 17</td>
</tr>
<tr>
<td>tst16</td>
<td>534.50 534.17 533.84 533.87 533.99</td>
<td>533.48 532.98 532.88 532.96 532.88</td>
</tr>
<tr>
<td></td>
<td>528 527 528 529 528</td>
<td>527 527 527 528 527</td>
</tr>
<tr>
<td></td>
<td>3 3 1 3 3</td>
<td>2 6 3 3 3</td>
</tr>
<tr>
<td>tst17</td>
<td>2460.27 2459.39 2459.25 2459.14 2459.20</td>
<td>2457.76 2457.35 2457.21 2456.91 2456.85</td>
</tr>
<tr>
<td></td>
<td>2450 2450 2450 2450 2450</td>
<td>2450 2450 2450 2450 2450</td>
</tr>
<tr>
<td></td>
<td>3 2 1 4 4</td>
<td>3 9 4 6 13</td>
</tr>
<tr>
<td>tst18</td>
<td>1530.29 1529.59 1529.38 1529.42 1529.69</td>
<td>1527.57 1527.39 1526.88 1527.32 1526.97</td>
</tr>
<tr>
<td></td>
<td>1521 1521 1521 1521 1521</td>
<td>1521 1521 1521 1521 1521</td>
</tr>
<tr>
<td></td>
<td>10 18 22 22 16</td>
<td>31 36 46 39 43</td>
</tr>
<tr>
<td>tst19</td>
<td>1020.82 1020.27 1020.01 1020.00 1020.01</td>
<td>1019.19 1018.69 1018.77 1018.42 1018.61</td>
</tr>
<tr>
<td></td>
<td>1013 1013 1012 1012 1013</td>
<td>1012 1012 1012 1012 1012</td>
</tr>
<tr>
<td></td>
<td>2 1 1 1 1</td>
<td>1 1 1 1 4</td>
</tr>
<tr>
<td>tst20</td>
<td>666.81 666.33 666.33 666.35 666.21</td>
<td>665.68 665.28 665.18 665.22 665.28</td>
</tr>
<tr>
<td></td>
<td>660 662 661 660 661</td>
<td>660 659 661 661 659</td>
</tr>
<tr>
<td></td>
<td>1 6 2 1 4</td>
<td>1 1 9 4 1</td>
</tr>
</tbody>
</table>
Table 9. Score of initial solution for different initialization methods for set 2 of real instances

<table>
<thead>
<tr>
<th>problem</th>
<th>random</th>
<th>upgma</th>
<th>g_j+greedy</th>
<th>n_j+greedy</th>
<th>n_j+fitch</th>
<th>heat</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANGI</td>
<td>425.80</td>
<td>240.00</td>
<td>234.00</td>
<td>229.42</td>
<td>226.00</td>
<td>226.00</td>
</tr>
<tr>
<td>CARP</td>
<td>1546.92</td>
<td>628.00</td>
<td>592.00</td>
<td>591.18</td>
<td>597.00</td>
<td>591.18</td>
</tr>
<tr>
<td>ETHE</td>
<td>898.58</td>
<td>399.00</td>
<td>385.00</td>
<td>386.32</td>
<td>382.00</td>
<td>382.00</td>
</tr>
<tr>
<td>GOLO</td>
<td>903.82</td>
<td>567.00</td>
<td>535.00</td>
<td>534.16</td>
<td>533.00</td>
<td>533.00</td>
</tr>
<tr>
<td>GRIS</td>
<td>476.50</td>
<td>200.00</td>
<td>195.00</td>
<td>184.48</td>
<td>186.00</td>
<td>182.22</td>
</tr>
<tr>
<td>ROPA</td>
<td>731.28</td>
<td>363.00</td>
<td>353.00</td>
<td>346.00</td>
<td>347.00</td>
<td>345.72</td>
</tr>
<tr>
<td>SCHU</td>
<td>2627.02</td>
<td>880.00</td>
<td>842.00</td>
<td>812.64</td>
<td>805.00</td>
<td>803.22</td>
</tr>
<tr>
<td>TENU</td>
<td>1604.18</td>
<td>748.00</td>
<td>728.00</td>
<td>719.00</td>
<td>719.00</td>
<td>710.40</td>
</tr>
</tbody>
</table>

Table 10. Performance comparison for the initialization methods for tst10 to tst20

<table>
<thead>
<tr>
<th>effort</th>
<th>medium</th>
<th>large</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>reheat</td>
<td>0 1 2 3 4</td>
<td>0 1 2 3 4</td>
<td>0 1 2 3 4</td>
</tr>
<tr>
<td>random</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>greedy</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n_j+fitch</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n_j</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n_j+greedy</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>allmethods</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

carefully determined, through the use of a tuning methodology based on Combinatorial Interaction Testing [8], to yield the best solution quality in a reasonable computational time.

An extensive experimentation was conducted to investigate the performance of SAMPARS over a set of 20 well known benchmark instances. In these experiments our algorithm was carefully compared with an existing simulated annealing implementation (LVB) [5, 6], and other two state-of-the-art algorithms. The results show that there exists a statistically significant increase in performance achieved by SAMPARS with respect to LVB. SAMPARS is in fact able to consistently improve the best results produced by LVB, obtaining in certain instances important reductions in the parsimony cost. Regarding GA+PR+LS [31], we have observed that in average this algorithm returns worse
Simulated Annealing Applied to the Resolution of Phylogenetic ... 93

quality solutions than SAMPARS. Compared with the state-of-the-art algorithm called Hydra [18], our SAMPARS algorithm was able to improve on 9 previous best-known solutions and to equal these results on the other 11 selected benchmark instances. Furthermore, it was observed that the solution cost found by SAMPARS presents a relatively small standard deviation, which indicates the precision and robustness of the proposed approach. These experimental results confirm the practical advantages of using our algorithm for solving the MP problem.

Finding near-optimal solutions for the MP problem is a very challenging problem. However, the introduction of SAMPARS opens up an exciting range of possibilities for future research. One fruitful possibility is to analyze the use of different cooling schedules, stop conditions and mechanism for adapting the maximum number of visited neighboring solutions at each temperature depending on the behavior of the search process.

References


Simulated Annealing Applied to the Resolution of Phylogenetic...


Complimentary Contributor Copy


Simulated Annealing Applied to the Resolution of Phylogenetic ...


Complimentary Contributor Copy
Chapter 5

**IMPROVED PARALLEL SIMULATED ANNEALING APPLIED TO SPACECRAFT TRAJECTORY OPTIMIZATION**

*Ya-Zhong Luo*¹ and *Guo-Jin Tang*²

College of Aerospace Science and Engineering, National University of Defense Technology, Changsha, China

**ABSTRACT**

Combining the advantages of simulated annealing (SA) with that of the simplex method, a new kind of parallel simulated annealing using the simplex method (PSASM) is proposed. PSASM is first compared with other popular stochastic optimization algorithms through the solution of several mathematical example problems in which it obtains the best-known solutions for two benchmark nonlinear constrained optimization problems, and with the lowest computational cost. PSASM is then applied to two representative types of spacecraft trajectory optimization problems. The first is a low-thrust Earth-to-Mars orbit transfer problem. The optimization results show that PSASM is superior to sequential SA, genetic algorithms, and the classical optimization algorithms. The second problem is the impulsive rendezvous trajectory optimization. A new optimization approach combining primer vector theory with PSASM for a

---

¹ Professor, luoyz@nudt.edu.cn.
² Professor, tanggj@nudt.edu.cn

Complimentary Contributor Copy
fuel-optimal non-linear impulsive rendezvous is proposed. PSASM is demonstrated to be able to guarantee solutions that satisfy Lawden’s necessary optimality conditions. These two applications show that PSASM is a successful global optimizer for spacecraft trajectory optimization design.

**INTRODUCTION**

The simulated annealing (SA) algorithm is a stochastic optimization algorithm originally proposed by Kirkpatrick et al. [1] for solving combinatorial problems. Its ideas stem from the comparability between solving optimization problems and physical annealing. Numerous researchers have demonstrated that SA is very effective in many optimization problems. However, SA’s long execution time has been a major drawback in practice. One method that is widely used to improve its performance is to parallelize SA by dividing a Markov main chain into subchains on multiple processors, and several representatively parallel SAs using this method have been proposed [2–6]. Another used method is to hybridize SA with other local deterministic algorithms [7, 8]. The simplex method (SM) is a well-known deterministic search method proposed by Nelder and Mead [9]. Press et al. [10] proposed a new simplex simulated annealing algorithm by combining SA with SM. In Press’s new algorithm, SM is applied for the generation of a new design vector instead of for random generation, and this algorithm has been applied in a number of studies [11–13]. The SM search procedure is employed on a simplex consisting of n-1 vertices for an n-dimensional optimization problem, and therefore, SM can be regarded as a population-based algorithm with natural parallelism. Consequently, it would be highly desirable to parallelize SA using SM.

In this chapter, we present an improved edition of the hybrid algorithm combining SA with SM, namely parallel simulated annealing using the simplex method (PSASM). The approach presented here is related to the technique in [14]. SA has been applied successfully to solve a significant number of combinatorial optimization problems, such as the design of telephone networks and integrated circuits, test pattern generation, image restoration, and statistical optimization. Nevertheless, it has not yet been used widely for continuous optimization. In addition, SA has rarely been applied to spacecraft trajectory optimization. A hide-and-seek SA algorithm proposed by Belisle et al. [15] was applied to optimize a nonsmooth aircraft trajectory [16]
and a missile trajectory [17]. The development of numerical methods for spacecraft trajectory optimization has closely paralleled the exploration of space and the development of the digital computer [18]. The spacecraft trajectory optimization problem can be stated simply as the determination of a trajectory for a spacecraft that satisfies specified initial and terminal conditions and that is conducting a required mission, while minimizing some quantity of importance including propellant cost, flight time, and the like [19]. The methods for numerically solving trajectory optimization problems are classified as either indirect or direct. A good survey of these methods can be found in [18-21]. No matter which type of method is employed, a spacecraft trajectory optimization problem is always formulated as a nonlinear programming problem. The gradient-based algorithms, such as Newton’s method or Sequential Quadratic Programming (SQP), are the most widely used optimizers. A qualitatively different approach, recently applied to spacecraft trajectory optimization, is the use of evolutionary algorithms [19]. In recent years, representative evolutionary algorithms, including genetic algorithms [22-23], differential evolution algorithms [24], particle swarm algorithms [25], and ant colony algorithms [26], among others, have been successfully applied to the solution of different spacecraft trajectory optimization problems. In this chapter, we show that PSASM is a successful global optimizer for spacecraft trajectory optimization design. Two representative types of spacecraft trajectory optimization problem will be investigated, including low-thrust trajectory optimization and impulsive rendezvous trajectory optimization. The chapter begins by introducing the concept and procedure of the PSASM algorithm. Then, four mathematical examples are used to test PSASM and compare it with other popular stochastic optimization algorithms. Applications of PSASM in spacecraft trajectory optimization are provided subsequently.

**PARALLEL SIMULATED ANNEALING USING THE SIMPLEX METHOD**

2.1. Simulated Annealing and the Simplex Method

The SA algorithm was developed originally for discrete combinatorial optimization problems, in which the design vector \( \mathbf{x} \) has finite or countably infinite configurations. The SA algorithm essentially imitates physical
annealing, a process by which a solid is first heated until it melts and then gradually cools until it crystallizes into a state with a perfect lattice. The final state that this process attains is a configuration that minimizes the free energy of the solid. At any given temperature $T$, the probability of a system in a state $r$ is given by the Boltzmann distribution

$$P(r) \sim e^{-E(r)/k_BT}$$

(1)

where $E(r)$ is the energy associated with state $r$, and $k_B$ is Boltzmann’s constant. Thus, at equilibrium, the most probable state is that of lowest energy. SA uses this principle of physical annealing. The state of a system corresponds to the design vector $x$, and its energy to the value of the cost function $f(x)$. The Boltzmann distribution is replaced by the Metropolis criterion

$$\beta^T = \min \left\{ 1, \exp \left( \frac{f(x_1) - f(x_2)}{T} \right) \right\}$$

(2)

where $x_1$ and $x_2$ are two different design points. SA avoids being trapped in local optima by probabilistically accepting transitions corresponding to deterioration in cost function value. This deterioration makes it possible to move away from local optima and explore the feasible region in its entirety. As the optimization process progresses, the probability of accepting cost function deteriorations slowly reduces to zero.

SA is an all-purpose algorithm that can be employed to solve any optimization problem. It is composed of three functions, a neighborhood search function, a new solution accepted function, and a cooling schedule, and two rules, the algorithm termination rule and the Metropolis sampling stabilization rule.

The Simplex Method (SM) is a robust nonlinear multi-dimensional optimization technique. The method does not require the derivatives of the function to be optimized. A simplex is a geometrical figure of $(n+1)$ vertices in $n$ dimensions. SM does not start with a single point, but with an initial simplex $(n+1)$ points; then, through a sequence of elementary geometric transformations (reflection, contraction, and extension), the initial simplex moves, expands, and contracts in such a way that it adapts itself to the function landscape and finally surrounds the optimum. For determining the appropriate transformation, the method uses only the relative orders between the
performances (values of the function to be optimized) of the points considered. After each transformation, the current worst point is replaced by a better one; in this way, provided certain precautions be taken, the algorithm always forces convergence of the sequence of iterates.

Although SM has a very wide application range because of its simplicity and efficiency, it is very sensitive to the initial guessed solution and always gets trapped at local minimum points. However, it should be remembered that SM has a unique characteristic, its natural parallelism. In stark contrast to other classical optimization algorithms, such as the Powell and gradient-based algorithms, the SM iteration process is based on a simplex including \( n+1 \) points, making SM easily parallelizable.

Given these descriptions, we can conclude that SA is a point-based, sequential, stochastic, global optimization algorithm, while SM is a population-based, parallel, determinate, local optimization algorithm. Because SA and SM have opposite characteristics, it can be expected that a method combining SA with SM would perform better than either SA or SM alone. One such combination is attempted here, and thus, a new approach, parallel simulated annealing using the simplex method (PSASM), is proposed.

2.2. Optimization Procedure of PSASM

The PSASM optimization procedure is described in Figure 1. A more detailed outline is as follows.

Step 1: A simplex (including \( n+1 \) design points) and other parameters, such as the initial temperature \( T_0 \) and the length of Metropolis sampling \( L_p \), are initialized, with \( k=1 \).

Step 2: The simplex is evaluated, and the highest, second-highest, and lowest points are determined.

Step 3: A new simplex is produced using reflection, contraction, and expansion.

Step 4: Simulated annealing search is applied to the \( i^{th} \) point of the simplex (to be referred to as \( x^i \)) with probability \( P(i, k) \) (described as (3)), \( i=1,2,...,n_0 \).

Step 4.1: Set the initial solution for Metropolis sampling at the current temperature \( T_k : y^i_l = x^i, l=1; \)
Step 4.2: Determine the tentative solution $y'_{i+1}$ using the neighborhood search function;

Step 4.3: Replace $y'_i$ with $y'_{i+1}$ using the Metropolis rule, $l+ = 1$;

Step 4.4: If $l > L_p$, $x'_i = y'_i$, go to Step 5; otherwise, go to Step 4.1;

Step 5: PSASM terminates if the stopping condition is satisfied; otherwise, $k = k + 1$, $T_k$ is renewed, go to Step 2.

PSASM combines the advantages of SM with those of SA, so it performs better than SM or SA alone, for the following reasons:

1) Hybridizing Different Search Mechanisms
SM is a determinate descent local search method, while SA is a stochastic global search method. Hybridizing such very different methods improves PSASM’s search ability and efficiency.

2) Parallel Search
Applying simulated annealing search at the points of the simplex not only parallelizes the Metropolis sampling at the current temperature, but also commutes sufficiently the results obtained by parallel simulated annealing search through the reflection, contraction and extension of the SM, making the parallel simulated annealing more efficient and effective. Furthermore, this method gives PSASM large parallelism granularity, so practical parallel computation performance, such as speedup, is improved substantially.

3) Probability for Employing Simulated Annealing on Simplex
A simplex can be regarded as a population including $n+1$ individuals, as those concepts are used in connection with genetic algorithms (GA). Therefore, simulated annealing search at points of the simplex is similar to the mutation operation of a GA, enabling an individual point of the simplex to break away from a local trap. Because a suitable individual point of the simplex can spread out using one of the operations, reflection, expansion, and contraction, it is not necessary to apply simulated annealing search on all points of the simplex. A probabilistic strategy must be applied to decide how and when simulated annealing search is employed at a point, just as a mutation probability is used in a GA. Here, a preliminary probabilistic strategy is proposed as follows:
Choose \( n_0 (n_0 \leq n + 1) \) points from the simplex at random. The probability function of the \( i^{th} (i = 1, 2, \cdots, n) \) point chosen to employ simulated annealing search on is

\[
P(i, k) = \exp\left(-\frac{k}{K_{\text{max}}}\right)
\]

where \( k \) is the annealing iteration index, and \( K_{\text{max}} \) is the maximum number of annealing iterations specified by the user. This strategy resembles a GA adaptive mutation strategy. As individual points are chosen with equal probabilities, the diversity of individual points is not reflected. During the SM prophase, more simulated annealing search is required to break away from local traps; while during the anaphase, simulated annealing search may be useless for obtaining the exact solution. Therefore, the probability function is designed to reduce the annealing.

### 2.3. Design of Parameters and Operations

#### 1) Neighborhood Search Function

One critical factor influencing SA performance is its neighborhood search function, whose purpose is to produce a tentative solution. The number of neighborhood states for a continuous variable is infinite, in contrast to the finite number for a combinational variable. One substantial difficulty in applying SA to the solution of continuous optimization problems is how to design an appropriate neighborhood search function. With the GA mutation strategy used here for reference, a new neighborhood search function is proposed as follows:

Given the current solution \( \mathbf{x} = (x_1, x_2, \cdots, x_n) \), choose \( x_{r1}, x_{r2}, \cdots, x_{rm} \) by random disturbance. Then, determine a tentative solution \( \mathbf{y} = (y_1, y_2, \cdots, y_n) \), all of whose elements other than \( y_{ri} (i = 1, 2, \cdots, m) \) are the same those of \( \mathbf{x} \):

\[
y_{ri} = x_{ri} + r \ast S \ast (x_{ri}' - x_{ri}), \quad i = 1, 2, \cdots, m
\]
where $r$ is a random number uniformly distributed between -1 and 1, $x_{i}^{U}$ and $x_{i}^{D}$ are the upper and lower bounds, respectively, of the $i^{th}$ design variable. $S$ and $m$ are two coefficients for scaling the neighborhood space. $m$ reflects the capacity for transverse exploitation of the search space; it should be a fixed
constant for holding the same neighborhood for each variable. Based on our
experiments, \( m = 1 \) is always effective and, in general, it can be determined
according to the dimension of the variables. \( S \) reflects the capacity for
longitudinal exploitation of the search space; it may be a constant or a variable
changing similarly to annealing. During the SA search prophase, it should be
large, so that SA can search in large-scale space and locate a global solution,
but during the anaphase, \( S \) should be small so that a rapid convergence to an
exact solution can be obtained. Therefore, the following function for \( S \) is
proposed.

\[
S = S_0 \exp(-b \cdot k / K_{\text{max}})
\]  

(5)

where \( S_0 \) is a constant in [0.05,0.5], and \( b \) is a coefficient determining the
nonlinear change trend, with \( b = 1 \) herein. According to the simulation
experiments, \( S \) described as (5) can improve SA performance effectively in
solving most optimization problems, while \( S \) as a constant improves SA
performance for some special optimization problems.

Considering the boundary constraints in practical problems, the disposal-
to-boundary constraints are as follows.

\[
y_r = \begin{cases} y_r, & x^p \leq y_r \leq x^u \\ x^p + p \cdot (x^u - x^p), & y_r > x^u \text{ or } y_r < x^p \end{cases}
\]  

(6)

where \( p \) is a random number uniformly distributed between zero and one.

This neighbor search function makes SA search effective for the following
reasons. (1) All solutions have the same neighborhood space, and new
solutions are produced randomly with equal probabilities, enabling SA to
search through the entire solution space. (2) Random disturbances at multiple
points enable every variable to reach the minimum. (3) Reducing the
neighborhood search space through annealing achieves an appropriate tradeoff
between large-scale global search and elaborate local search and effectively
accelerates the convergence rate. (4) A probability-based disposal-to-boundary
constraint is simple and enriches the diversity of Metropolis sampling.
2) Initial Temperature and Cooling Schedule

The initial temperature must be sufficiently large to ensure that the initial tentative solutions are accepted with a high probability, and this can be determined by the best and the worst objective functions of a large number of tentative solutions. In this study, a stochastic population is produced and evaluated, with \( f_{\text{min}}, f_{\text{max}} \) as the best and the worst objective functions, respectively. Then, \( T_0 \) is calculated

\[
T_0 = \frac{(f_{\text{max}} - f_{\text{min}})}{\ln(P_0)}
\]  

(7)

where \( P_0 \) is the initial acceptance probability for the worst solution. Therefore, \( T_0 \) is adjusted by \( P_0 \) and by the comparative characteristics of the initial population. The temperature schedule is taken to be

\[
T_{k+1} = cT_k, \quad 0.7 < c < 0.99
\]  

(8)

where \( c \) is an annealing coefficient. This temperature schedule is an exponential one, tested to be effective in encompassing both optimization quality and time consumption.

3) Metropolis Sampling Stabilization Rule

The Metropolis sampling stabilization rule is used to determine the number of tentative solutions at a given temperature. We choose a fixed step number as the stabilization rule, i.e., a fixed number of tentative solutions are produced at any given temperature and then Metropolis sampling terminates. For SA, this number should be large; a smaller number will be effective for PSASM. For clarity, we use \( L_p \) and \( L_s \) to denote this number for PSASM and SA, respectively.

4) Stopping Condition

The PSASM stopping condition can be based on that of either SM or SA. Here we consider only the one for SA. We choose \( K_{\text{max}} \) as a stopping condition; if \( k \geq K_{\text{max}} \), PSASM terminates.
2.4. Constrained Optimization

The basic SA and SM algorithms are defined for unconstrained problems. Because most engineering problems are constrained in one way or another, it is important to add the ability to deal with constraints. A penalty function is introduced for this purpose. For the general constrained problem

\[ \min f(x) \]
\[ \text{s.t. } g_i(x) \leq 0, \quad i = 1, 2, \ldots, p \]
\[ h_j(x) = 0, \quad j = 1, 2, \ldots, q \]  

the new, penalized, objective function \( F(x,M) \) is

\[
F(x,M) = f(x) + M\sum_{i=1}^{p} \max(0, g_i(x)) + \sum_{j=1}^{q} |h_j(x)| 
\]  

**MATHEMATICAL EXAMPLE PROBLEM**

To evaluate PSASM performance, we choose a set of test functions including two unconstrained functions and two constrained functions previously used to test other types of stochastic algorithms [27-39]. Herein, performance is measured by both the best solution obtained by the test algorithms and the number of function evaluations (NFE) needed to obtain the global or best solution. One unconstrained complex function is a two-dimensional function, and the other is a high-dimensional Rastrigin function. The two constrained functions include a five-dimensional one and Golinski’s speed reducer problem.

3.1. Nonlinear Unconstrained Function

**Test Function 1**

A two-dimensional unconstrained function is selected as the first example,

\[
\min f_1(x) = \frac{1}{2}(x_1^2 + x_2^2) - \cos(20\pi x_1)\cos(20\pi x_2) + 2, -10 < x_i < 10
\]  

Complimentary Contributor Copy
This function has only two design variables but is nonseparable and highly multimodal with 40,000 local minima in the range [-10,10]. The global minimum is located at $\mathbf{x}^* = (0,0)^T$, with the optimal objective function value $f_1(\mathbf{x}^*) = 1$. This function is tested with a precision of $|f_i(\mathbf{x}) - f_i(\mathbf{x}^*)| \leq 0.0001$. It has been solved using different stochastic optimization algorithms, including generalized extremal optimization (GEO) [27], region reduction SA (RPSA) [28], simulated evolution (SE) [29], and guided evolutionary SA (GES) [29]. SA and PSASM are used individually to optimize this function with 50 execution runs for each algorithm. The SA parameters are $K_{max} = 300$, $L_s = 300$, and $\alpha = 0.95$, and the PSASM parameters are $K_{max} = 300$, $n_0 = 2$, $L_p = 50$, and $\alpha = 0.95$. The results are listed in Table 1, together with the results obtained using SE, GESA, PRSA, and GEO reported by [27-29] for comparison.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[1.0000, 1.0001]</td>
<td>66%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>[1.0024, 1.0026]</td>
<td>34%</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Average NFE</td>
<td>800,000</td>
<td>800,000</td>
<td>92,254</td>
<td>164,070</td>
<td>57,452</td>
<td>5,969</td>
</tr>
</tbody>
</table>

As shown in the table, PSASM clearly demonstrates better performance than SE, GESA, PRSA, GEO, and SA. PSASM locates the global solution with 100% probability, and its computational cost is the lowest, only $\frac{1}{130} \sim \frac{1}{10}$ of those of the other algorithms.

**Test Function 2 (Rastrigin Function)**

The analytical Rastrigin Function is

$$
\text{min } f_2(\mathbf{x}) = \sum_{i=1}^{n} [x_i^2 - 10 \cos(2\pi x_i) + 10], \quad |x_i| \leq 5.12 \quad (12)
$$

The optimal objective function is zero. This function is non-convex and has many local minima distributed irregularly. These characteristics make it very difficult to get the optimal solution effectively and reliably. A 30-
dimensional \((n = 30)\) version is used for numerical simulation. The aim here is
to show that PSASM is successful even for such a high-dimensional function.
PSASM’s performance is compared with that of a standard floating-coded GA
[30] (arithmetical crossover operator, non-uniform mutation operator,
proportion select operator). Thirty stochastic calculations are conducted for
each algorithm. In order to analyze the influence of the parameters \(n_0\) and \(L_p\)
on optimization performance, five PSASM configurations with different \((n_0, \ L_p)\) values are conducted.

A precision of \(|f_j(x) - f_j(x^*)| \leq 0.1\) is adopted, and the PSASM NFE for
locating this solution is recorded. The statistical results are listed in Table 2.
The GA parameters are population size: 100, maximum generation number:
800, crossover probability: 0.92, and mutation probability: 0.10. The PSASM
parameters are \(K_{\text{max}} = 800\) and \(\alpha = 0.85\).

| Table 2. Results of PSASM with different \(n_0\) and \(L_p\) values
for a 30-dimensional Rastrigin Function |
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PSASM with different (n_0, L_p)</td>
</tr>
<tr>
<td>-----------------------------------------</td>
</tr>
<tr>
<td>(n_0 = 2, L_p = 100)</td>
</tr>
<tr>
<td>(n_0 = 3, L_p = 100)</td>
</tr>
<tr>
<td>(n_0 = 3, L_p = 67)</td>
</tr>
<tr>
<td>(n_0 = 4, L_p = 100)</td>
</tr>
<tr>
<td>(n_0 = 4, L_p = 50)</td>
</tr>
</tbody>
</table>

The convergence histories of a standard GA and the PSASMs are
compared in Figure 2. It can be seen from the figure that the PSASMs
evidently outperform the standard GA. The GA fails to get the optimal
solution. In contrast, three PSASMs with different parameters obtain the near-optimal solution in all experiments, showing that PSASM is robust.

As Table 2 demonstrates, when \(L_p\) is fixed as 100, \(n_0 = 3\) and \(n_0 = 4\)
require a little larger NFE than \(n_0 = 2\). In addition, when \(n_0\) is fixed at 3 or 4, a
slight larger \(L_p\) results in an increase in NFE. Thus, it can be concluded that
only a small \(n_0\) and \(L_p\) can be efficient and effective when PSASM is
implemented using only a single computer. However, it should be mentioned
that PSASM can be implemented easily using a cluster of computers. As the
total NFEs of \(n_0 = 2, 3\) and 4 are close, it can be expected that optimizing this
function using four computers would take less time than using two or three computers.

Figure 2. Convergence history of GA and PSASMs.

3.2. Nonlinear Constrained Function

Test Function 3

A constrained mathematical function with five design variables is considered.

\[
\min f_3(x) = 5.3578547x_1^2 + 0.8356891x_4x_5 + 37.293239x_1 - 40792.141 \\
s.t. 0 \leq c_1(x) \leq 92, 90 \leq c_2(x) \leq 110, 20 \leq c_3(x) \leq 25 \\
78 \leq x_1 \leq 102, 33 \leq x_2 \leq 45, 27 \leq x_3, x_4, x_5 \leq 45
\]

(13)
Improved Parallel Simulated Annealing …

where
\[
\begin{align*}
    & c_1(x) = 85.334407 + 0.0056858 x_2 x_3 + 0.0006262 x_4 x_5 - 0.0022053 x_1 x_5 \\
    & c_2(x) = 80.51249 + 0.0071317 x_2 x_5 + 0.0029955 x_2 x_3 + 0.0021813 x_3^2 \\
    & c_3(x) = 9.300961 + 0.0047026 x_3 x_5 + 0.0012547 x_1 x_4 + 0.0019085 x_3 x_4
\end{align*}
\]

This problem has been solved using the chaos optimization algorithm (COA) [31], GA [32], and evolutionary programming (EP) [33].

The SA parameters are \( K_{\text{max}} = 200 \), \( L_s = 50 \), and \( \alpha = 0.95 \), and the PSASM parameters are \( K_{\text{max}} = 300 \), \( L_p = 10 \), \( n_0 = 2 \), and \( \alpha = 0.95 \). The best solutions obtained by SA and PSASM during ten successive runs are presented in Table 3, together with the solutions reported in [31-33].

Table 3 illustrate clearly that PSASM is capable of solving this constrained problem successfully using just the very simple penalty function described in (10). PSASM obtains the best-known solution for this problem. Of the other methods, only EP gets a close solution. However, the average objective function value obtained by PSASM is \(-31,015\), while EO obtained only \(-31,006.2 \) [30]. Table 3 also presents the average NFE of the different algorithms. The GA, EP, and SA data are estimated approximately using stopping conditions. The COA data were reported by [31]. The PSASM data are the statistical result of giving the objective function a stopping condition of \(-31,010\). Judging from the computational cost, PSASM is the most efficient. Figure 3 compares the iteration process of SA and PSASM. It is very clear that PSASM has a faster convergence rate and a better solution.

Homaifar et al. [32] first used this problem as a test function. It was introduced originally in Himmelblau’s book [34], in which the optimal solution was confidently claimed to be

\[
x^* = (78,33,29.995256025682,45,36.7758129057), f(x^*) = -30,665.539
\]

Table 3. Best solutions of Test Function 3 obtained using different algorithms

<table>
<thead>
<tr>
<th></th>
<th>( f_3(x) )</th>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( x_3 )</th>
<th>( x_4 )</th>
<th>( x_5 )</th>
<th>( c_1(x) )</th>
<th>( c_2(x) )</th>
<th>( c_3(x) )</th>
<th>Average NFE</th>
</tr>
</thead>
<tbody>
<tr>
<td>COA [31]</td>
<td>-30,862.53</td>
<td>78.63</td>
<td>33.26</td>
<td>27.79</td>
<td>44.26</td>
<td>43.49</td>
<td>91.80</td>
<td>100.34</td>
<td>20.07</td>
<td>10,198</td>
</tr>
<tr>
<td>GA [32]</td>
<td>-30,175.80</td>
<td>80.61</td>
<td>34.21</td>
<td>31.34</td>
<td>42.05</td>
<td>34.85</td>
<td>90.58</td>
<td>99.41</td>
<td>20.12</td>
<td>40,000</td>
</tr>
<tr>
<td>EP [33]</td>
<td>-31,022.65</td>
<td>78.00</td>
<td>33.00</td>
<td>27.08</td>
<td>45</td>
<td>44.97</td>
<td>92.00</td>
<td>100.40</td>
<td>20.01</td>
<td>4,000</td>
</tr>
<tr>
<td>SA</td>
<td>-31,004.98</td>
<td>78.00</td>
<td>33.10</td>
<td>27.23</td>
<td>44.92</td>
<td>44.53</td>
<td>91.95</td>
<td>100.38</td>
<td>20.01</td>
<td>10,000</td>
</tr>
<tr>
<td>PSASM</td>
<td>-31,025.02</td>
<td>78.00</td>
<td>33.00</td>
<td>27.07</td>
<td>44.99</td>
<td>44.96</td>
<td>92.00</td>
<td>100.40</td>
<td>20.00</td>
<td>3,315</td>
</tr>
</tbody>
</table>
However, from the results reported in [31-33] and our study, it is clear that the optimal objective function value is less than $-30,665.539$. Which is wrong? In fact, Homaifar et al., made a citation mistake. The original formula of $c_1(x)$ in [34] is

$$c_1(x) = 85.334407 + 0.0056858x_2x_3 + 0.0006262x_4x_4 - 0.0022053x_5x_5$$

The original coefficient of $x_4x_4$ is 0.0006262, but Homaifar et al., cited it improperly as 0.00026. Refs. [31-33] all cited Homaiifar’s improper mathematical formula. Therefore, solutions better than “the optimal solution” were obtained! Herein, the reference optimal solution obtained using PSASM (after 20,000 function evaluations) for Homaiifar’s problem are

$$x^* = (78.000107, 33.000102, 27.071213, 44.999100, 44.968919),$$
$$f(x^*) = -31,025.510716$$

Complimentary Contributor Copy
Test Function 4 (Golinski’s Speed Reducer Problem)

Golinski’s speed reducer problem is one of the most well studied problems in the NASA-Langley Multidisciplinary Design Optimization (MDO) Test Suite. Many researchers, for example, Rao [35], Li, and Papalambros [36], Kuang et al. [37], Azarm and Li [38], and Ray [39] have reported solutions to this problem. However, none of the reported solutions other than Ray’s is feasible, including the one that appears in the MDO Test Suite itself. This chapter presents the best-known feasible solution obtained by PSASM.

The objective of Golinski’s speed reducer problem is to find the minimum weight, subject to several constraints. There are seven design variables: width of the gear face $x_1$, teeth module $x_2$, number of pinion teeth $x_3$, shaft-one length between bearings $x_4$, shaft-two length between bearings $x_5$, diameter of shaft one $x_6$, and diameter of shaft two $x_7$. This objective leads to the following constrained optimization problem.

$$
\min \quad f_\delta(x) = 0.7854x_1x_2^2(3.3333x_2^2 + 14.9334x_3 - 43.0934) - 1.5079x_7(x_6^2 + x_7^2) + 7.477(x_6^2 + x_7^2) + 0.7854(x_4x_6^2 + x_5x_7^2)
$$

(14)

s.t. \quad g_i(x) \leq 1(i = 1,2,\ldots,11), \quad 2.6 \leq x_1 \leq 3.6, \quad 0.7 \leq x_2 \leq 0.8, \quad 13 \leq x_3 \leq 28\text{(integer value)},

\begin{align*}
7.3 \leq x_4 \leq 8.3, & \quad 7.3 \leq x_5 \leq 8.3, \quad 2.9 \leq x_6 \leq 3.9, \quad 5.0 \leq x_7 \leq 5.5
\end{align*}

(15)

where

\begin{align*}
g_1(x) = 27x_1^2x_2^3x_3, & \quad g_2(x) = 397.5x_1^2x_2^2x_3^2, \quad g_3(x) = 1.93x_1^2x_2^4x_3^2x_4^4, \quad g_4(x) = 1.93x_1^2x_2^3x_3^3x_4^3, \\
g_5(x) = \frac{[(745x_4x_2^3x_3^4)^2 + 16.9 \times 10^6]^\frac{1}{2}}{[110.0x_4^3]}, & \quad g_6(x) = \frac{[(745x_5x_2^3x_3^4)^2 + 157.5 \times 10^6]^\frac{1}{2}}{[85.0x_5^3]}, \quad g_7(x) = x_2x_3^4/40, \\
g_8(x) = 5x_2/x_4, & \quad g_9(x) = x_4/12x_2, \\
g_{10}(x) = (1.5x_6 + 1.9)x_4^2, & \quad g_{11}(x) = (1.1x_7 + 1.9)x_7^{-1}.
\end{align*}
Results of five successive runs are presented in Table 4. The best solution obtained using PSASM is compared with other reported results in Table 5. From Table 5, it is clear that the solutions reported by Rao [35], Li and Papalambros [36], Kuang et al. [37], Azarm and Li [38], and NASA are infeasible. Only Ray [39] obtained a feasible solution, using a swarm algorithm. The solution obtained by PSASM is better than Ray’s. In addition, Ray’s solution required 70,000 function evaluations, while PSASM located its solution after approximately 7,000 function evaluations, only one-tenth as many, clearly demonstrating the efficacy of PSASM.

Table 4. Results of five trials using PSASM

<table>
<thead>
<tr>
<th>Solution</th>
<th>Run 1</th>
<th>Run 2</th>
<th>Run 3</th>
<th>Run 4</th>
<th>Run 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>x₁</td>
<td>3.50001966</td>
<td>3.50004924</td>
<td>3.5000547</td>
<td>3.50019634</td>
<td>3.5000171</td>
</tr>
<tr>
<td>x₂</td>
<td>0.70000331</td>
<td>0.70000984</td>
<td>0.70000001</td>
<td>0.70000007</td>
<td>0.7000033</td>
</tr>
<tr>
<td>x₃</td>
<td>17</td>
<td>17</td>
<td>17</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>x₄</td>
<td>7.30001163</td>
<td>7.30000465</td>
<td>7.30001127</td>
<td>7.30000452</td>
<td>7.3000644</td>
</tr>
<tr>
<td>x₅</td>
<td>7.71532052</td>
<td>7.71616417</td>
<td>7.71534120</td>
<td>7.71539161</td>
<td>7.71532769</td>
</tr>
<tr>
<td>x₆</td>
<td>3.35021714</td>
<td>3.35046639</td>
<td>3.35021476</td>
<td>3.35021676</td>
<td>3.35021590</td>
</tr>
<tr>
<td>x₇</td>
<td>5.28665446</td>
<td>5.28665483</td>
<td>5.28673739</td>
<td>5.28671488</td>
<td>5.28665460</td>
</tr>
<tr>
<td>x₈</td>
<td>2.99437844</td>
<td>2.99450179</td>
<td>2.99437008</td>
<td>2.99447296</td>
<td>2.99435783</td>
</tr>
</tbody>
</table>

Table 5. Results of speed reducer design

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>x₁</td>
<td>3.500000</td>
<td>3.500000</td>
<td>3.500000</td>
<td>3.500000</td>
<td>3.500000</td>
<td>3.500000</td>
<td>3.500000</td>
</tr>
<tr>
<td>x₂</td>
<td>0.700000</td>
<td>0.700000</td>
<td>0.700000</td>
<td>0.700000</td>
<td>0.700000</td>
<td>0.700000</td>
<td>0.700000</td>
</tr>
<tr>
<td>x₃</td>
<td>17</td>
<td>17</td>
<td>17</td>
<td>17</td>
<td>17</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>x₄</td>
<td>7.300000</td>
<td>7.300000</td>
<td>7.300000</td>
<td>7.300000</td>
<td>7.300000</td>
<td>7.300000</td>
<td>7.300000</td>
</tr>
<tr>
<td>x₅</td>
<td>7.715320</td>
<td>7.716164</td>
<td>7.715341</td>
<td>7.715391</td>
<td>7.715327</td>
<td>7.715327</td>
<td>7.715327</td>
</tr>
<tr>
<td>x₇</td>
<td>5.286654</td>
<td>5.286654</td>
<td>5.286737</td>
<td>5.286714</td>
<td>5.286654</td>
<td>5.286654</td>
<td>5.286654</td>
</tr>
<tr>
<td>x₈</td>
<td>2.994378</td>
<td>2.994501</td>
<td>2.994370</td>
<td>2.994472</td>
<td>2.994357</td>
<td>2.994357</td>
<td>2.994357</td>
</tr>
<tr>
<td>g₁</td>
<td>0.926084</td>
<td>0.926084</td>
<td>0.926084</td>
<td>0.926084</td>
<td>0.926084</td>
<td>0.926084</td>
<td>0.926084</td>
</tr>
<tr>
<td>g₂</td>
<td>0.802015</td>
<td>0.802015</td>
<td>0.779723</td>
<td>0.802001</td>
<td>0.802001</td>
<td>0.802001</td>
<td>0.802001</td>
</tr>
<tr>
<td>g₃</td>
<td>0.500956</td>
<td>0.500956</td>
<td>0.472138</td>
<td>0.500956</td>
<td>0.500827</td>
<td>0.500827</td>
<td>0.500827</td>
</tr>
<tr>
<td>g₄</td>
<td>0.080566</td>
<td>0.094918</td>
<td>0.123144</td>
<td>0.094918</td>
<td>0.080779</td>
<td>0.095283</td>
<td>0.095363</td>
</tr>
<tr>
<td>g₅</td>
<td>1.000192</td>
<td>1.000192</td>
<td>0.956711</td>
<td>1.000192</td>
<td>1.000000</td>
<td>1.000000</td>
<td>0.999999</td>
</tr>
<tr>
<td>g₆</td>
<td>0.998026</td>
<td>0.998109</td>
<td>1.182060</td>
<td>0.998109</td>
<td>1.000000</td>
<td>1.000000</td>
<td>0.999999</td>
</tr>
<tr>
<td>g₇</td>
<td>0.297500</td>
<td>0.297500</td>
<td>0.297500</td>
<td>0.297500</td>
<td>0.297500</td>
<td>0.297500</td>
<td>0.297500</td>
</tr>
<tr>
<td>g₈</td>
<td>1.000000</td>
<td>1.000000</td>
<td>0.972222</td>
<td>1.000000</td>
<td>1.000000</td>
<td>1.000000</td>
<td>0.999999</td>
</tr>
<tr>
<td>g₉</td>
<td>0.416667</td>
<td>0.416667</td>
<td>0.428571</td>
<td>0.416667</td>
<td>0.416667</td>
<td>0.416667</td>
<td>0.416667</td>
</tr>
<tr>
<td>g₁₀</td>
<td>0.94863014</td>
<td>0.94863014</td>
<td>0.95890411</td>
<td>0.94863014</td>
<td>0.94867419</td>
<td>0.94867413</td>
<td>0.94867366</td>
</tr>
</tbody>
</table>

Boldfaced type indicates the violated constraints.
4. LOW-THRUST TRAJECTORY OPTIMIZATION

It is well known that spacecraft propelled by low-thrust electric propulsion can potentially deliver a greater payload fraction compared to spacecraft propelled by conventional chemical propulsion. However, optimizing low-thrust orbit transfers is a challenging problem due to low control authority of the electric propulsion system and the existence of long powered arcs and subsequent multiple orbital revolutions [19]. Therefore, optimization design of low-thrust trajectory transfer has drawn increasing attention, and a number of approaches have been proposed [40-48]. In these approaches, gradient-based algorithms, such as Newton’s method and SQP, are the most widely used. In recent years, GAs have been used widely as well. The effectiveness of a GA in solving near-optimal low-thrust minimum-time Earth-to-Mars orbit transfer was investigated by Rauwolf and Coverstone-Carroll [40]. Furthermore, combining GA with the SQP algorithm, Crain et al. [43] and Hughes and McInnes [44] successfully designed a hybrid optimization approach for solving low-thrust orbit transfer problems. In contrast, SA was seldom applied in low-thrust orbit design [7]. In this part, we will show that PSASM is an effective and efficient optimizer for low-thrust trajectory design [49].

4.1. Problem Definition

The present study considers low-thrust trajectories from one planet to another. The problem under consideration is an orbital transfer from Earth to Mars with the orbit transfer geometry illustrated in Figure 4.

Canonical units are used to describe this problem. The units of time and distance are TU and AU, respectively, and the attracting center is the sun. The equations of motion for a low-thrust orbit transfer in a plane are written as follows [50].

\[
\begin{align*}
\dot{r} &= u \\
\dot{\theta} &= \frac{v}{r} \\
\dot{u} &= \frac{v^2}{r} - \frac{\mu}{r^2} + \frac{T \sin \varphi}{m_0 - \dot{m} t} \\
\dot{v} &= -\frac{uv}{r} + \frac{T \cos \varphi}{m_0 - \dot{m} t}
\end{align*}
\]

(16)

Complimentary Contributor Copy
Figure 4. Geometry of an Earth-to-Mars orbit transfer.

The initial conditions, \( r(0), \theta(0), u(0), v(0) \), correspond to the spacecraft’s position and velocity in the Earth’s orbit at the initiation of the transfer. The initial conditions for the considered problem are

\[
\begin{align*}
    r(0) & = 1.0, \\
    \theta(0) & = 0.0, \\
    u(0) & = 0.0, \\
    v(0) & = 1.0
\end{align*}
\]  

(17)

where \( r(0) \) is in AU, and both \( u(0) \) and \( v(0) \) are in AU/TU.

Similarly, the terminal boundary conditions corresponding to a circular Mars orbit are

\[
\begin{align*}
    r_K & = 1.524, \\
    u_K & = 0.0, \\
    v_K & = 0.810, \\
    \theta_K & \text{ free}
\end{align*}
\]  

(18)

Note that for orbit transfer problems, the final angular displacement is not specified, i.e., \( \theta_K \) is free.

For this test problem, the modeled propulsion system contains a fixed power source, such as a nuclear electric engine. The thrust- and mass-related terms are chosen to be consistent with the problem in [50]:

\[
\begin{align*}
    m_0 & = 4545.4 \text{ kg}, \\
    \dot{m} & = 6.787 e^{-5} \text{ kg/s}, \\
    T & = 3.787 \text{ N}
\end{align*}
\]  

(19)

The optimization goals of orbit transfer are to choose a thrust history, including thrust direction and magnitude. In order to simplify the spacecraft’s control system, the transfer trajectory is divided into a fixed-number of
trajectory segments when designing the transfer orbit from planet to planet. Only two variables per trajectory segment need to be optimized, the thrust angle and a variable indicating whether the engines are on or off. The second variable is called a switch. In our study, the trajectory is divided into ten segments, as in [40], and the total number of variables is 19, including ten angles and nine switches. Since the engines are required to be on during the first segment to indicate the transfer, only the thrust angle is of concern during this segment.

According to the above description, the design vector \( \mathbf{x} \) includes 19 variables: ten continuous variables and nine switch control variables

\[
\mathbf{x} = \{ \phi_1, \phi_2, \cdots, \phi_{10}, t_{w_2}, t_{w_3}, \cdots, t_{w_{10}} \}^T
\]

(20)

where \( \phi_i \in [-\pi, \pi] (i = 1, 2, \cdots, 10) \), \( t_{w_j} \in \{0,1\} (j = 2, 3, \cdots, 10) \). When \( t_{w_j} = 1 \), the engine is on; when \( t_{w_j} = 0 \), the engine is off. The real-coded method is used for continuous variables and the binary-coded method for switch control variables. A nine-bit string \( T_w = t_{w_1}t_{w_9} \cdots t_{w_3}t_{w_2} \) is used to describe the switch control variables as a whole. The search space of \( T_w \) is the binary space \( 000000000 - 111111111 \), with the corresponding integer space \( \hat{T}_w = \{ T_w | T_w \in Z, 0 \leq T_w \leq 511 \} \).

The objectives of orbit transfer optimization design include two classes of performance index. One is the minimum transfer time: \( \min t_f \), \( t_f \) being the terminal time of transfer. The other is the minimum fuel consumption for a fixed time transfer, with the performance index being equal to the maximum terminal mass: \( \max m(t_f) \), under the assumption that the initial mass of the spacecraft is determinate. In Bryson and Ho’s study [50], the optimal minimum transfer time \( t_f^* \) for the test problem was found to be 193 days. In this study, PSASM is used to solve not only the optimal minimum transfer time problem, for which the optimal solution is known, but also the optimal minimum fuel consumption problem with a fixed transfer time.

This transfer trajectory optimization design problem has equality constraints. We adopt several scaling methods to deal with the optimization objective and the equality constraints. Several factors are considered for scaling the problem, including that the error for equality constraints is
controlled within $\varepsilon = 1e-3$, the optimal transfer time is 193 days, and the terminal mass is about 3,000-4,000 kg.

After dealing with the constraints, the comprehensive optimization objective is

$$\min f_1(x) = M \left( \frac{t_f}{193} \right)^2 + \frac{(r(t_f) - r_k)^2}{0.001^2} + \frac{(u(t_f) - u_k)^2}{0.001^2} + \frac{(v(t_f) - v_k)^2}{0.001^2}$$

(21)

for minimum transfer time, and the comprehensive optimization objective after dealing with the constraints is

$$\max f_2(x) = -M \left( \frac{m(t_f)}{3000} \right)^2 + \frac{(r(t_f) - r_k)^2}{0.001^2} + \frac{(u(t_f) - u_k)^2}{0.001^2} + \frac{(v(t_f) - v_k)^2}{0.001^2}$$

(22)

for maximum final mass, where $M$ is a weight coefficient, obtained by trial and error, reflecting the importance of the optimization objective in the comprehensive function.

4.2. Numerical Optimization Results

1. Minimum Transfer Time

An assumption for minimum-time optimization is made based on the considerations of practical background and convenient application of the classical algorithms for solving this problem. The assumption is that the transfer trajectory contains no coasting segment, i.e., the engine is always on and the engine thrust directions are the only optimization variables.

Four different algorithms, Powell [51], SM, SA, and PSASM, are tested for solving the low-thrust Earth-to-Mars minimum-time orbit transfer problem. Generating the initial solution at random, ten experiments are conducted for each algorithm. The SA parameters are $T_0 = 10,000$, $K_{\text{max}} = 200$, $L_p = 400$, and $\alpha = 0.95$. The PSASM parameters are $K_{\text{max}} = 200$, $T_0 = 10,000$, $n_0 = 4$, $L_p = 200$, and $\alpha = 0.95$.

Each algorithm can obtain a feasible solution (satisfying the boundary constraints) in all experiments, indicating that the comprehensive objective function (21) is reasonable and effective.
Table 6 provides the statistical optimization results of the four algorithms. The best solutions obtained by SM, Powell, SA, and PSASM are summarized in Table 7, along with the best solution obtained by a GA in [40] and the analytical optimal solution for comparison.

As shown in Tables 6 and 7, it is very clear that low-thrust trajectory transfer is a very complex optimization problem with many local points. The classical algorithms, Powell and SM, always trap in a local minimum, far from the global minimum. Both SA and PSASM obtained near-optimal solutions in all experiments, with both outperforming the GA [40] in solving this low-thrust trajectory transfer problem, and the relative error from the optimal solution is only between 0.21%-3.06%. As seen from Tables 6 and 7, PSASM obtained a better solution than SA.

The MPI library is used to implement PSASM on parallel computers. The hardware environment is a cluster of computers (Dell Dimension 8300) with a 100M network. The theoretical analysis of PSASM speedup shows that when the optimization problem has a high time cost and the number of the computers is $n_0$, the speedup ($S_p = \frac{t_1}{t_p}$) is given as follows [52]:

$$S_p \approx \frac{L_p n_0 + \bar{m}}{L_p + \bar{m}}$$

(23)

where $\bar{m}$ is a statistical constant equal to about 2.5. The PSASM execution time on one computer ($t_1$) is about 3,659.0 s, and the execution time on four computers ($t_p$) is about 940.1 s, with $S_p$ calculated as 3.89. From (23), $S_p$ is about 3.96, so the experimental result is consistent with the theoretical result. PSASM has demonstrated an excellent parallel efficiency in solving the low-thrust trajectory transfer problem. The SA execution time is about 1,861.2 s. Thus, we can conclude that PSASM excels over the SA both in solution quality and time cost.

2. Minimum Fuel Consumption with Fixed Transfer Time

The essence of optimizing the minimum fuel consumption with fixed transfer-time is to get as many coasting trajectory segments as possible. Herein, we fix $t_f = 230$ days. SA and PSASM are used to solve the minimum-fuel-consumption optimization problem, with ten experiments.

Complimentary Contributor Copy
conducted for each algorithm. The SA and PSASM parameters are the same as those adopted in minimum-transfer-time optimization. SA has a success rate of 90%, with 100% for PSASM. The best terminal mass obtained by SA is 3,735.1 kg (the coasting time is 40% of the transfer time), with 3,870.0 kg (the coasting time is 50% of the transfer time) for PSASM. PSASM also outperforms SA in solving the minimum-fuel-consumption low-thrust trajectory transfer problem. The minimum fuel consumption optimization results of five successive PSASM runs are presented in Table 8. Figure 5 illustrates the history of the direction of thrust corresponding to the solution of “Run 4” in Table 8. As shown in the table, PSASM does not locate the best solution in all runs. Thus, improvement measures to make PSASM more robust need to be investigated.

Table 6. Results of Minimum-time Optimization

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Objective</th>
<th>Best</th>
<th>Worst</th>
<th>Mean</th>
<th>Standard deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>SM</td>
<td>238.2</td>
<td>345.1</td>
<td>295.4</td>
<td>42.00</td>
<td></td>
</tr>
<tr>
<td>Powell</td>
<td>217.4</td>
<td>326.8</td>
<td>266.1</td>
<td>37.47</td>
<td></td>
</tr>
<tr>
<td>SA</td>
<td>195.1</td>
<td>198.9</td>
<td>197.3</td>
<td>1.29</td>
<td></td>
</tr>
<tr>
<td>PSASM</td>
<td>193.4</td>
<td>197.4</td>
<td>195.0</td>
<td>1.13</td>
<td></td>
</tr>
</tbody>
</table>

Table 7. Best Solution of Different Algorithms for Minimum-time Optimization

<table>
<thead>
<tr>
<th></th>
<th>$t_f$, day</th>
<th>$r(t)$, AU</th>
<th>$\Theta(t)$, deg</th>
<th>$u(t)$, AU/TU</th>
<th>$v(t)$, AU/TU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimal</td>
<td>193</td>
<td>1.524</td>
<td>free</td>
<td>0.000</td>
<td>0.810</td>
</tr>
<tr>
<td>SA</td>
<td>195.1</td>
<td>1.524</td>
<td>143.6</td>
<td>0.000</td>
<td>0.809</td>
</tr>
<tr>
<td>PSASM</td>
<td>193.4</td>
<td>1.524</td>
<td>142.2</td>
<td>0.000</td>
<td>0.810</td>
</tr>
<tr>
<td>GA [40]</td>
<td>199</td>
<td>1.512</td>
<td>149.5</td>
<td>0.008</td>
<td>0.802</td>
</tr>
<tr>
<td>Powell</td>
<td>217.4</td>
<td>1.524</td>
<td>163.1</td>
<td>0.001</td>
<td>0.809</td>
</tr>
<tr>
<td>SM</td>
<td>238.2</td>
<td>1.524</td>
<td>176.9</td>
<td>0.001</td>
<td>0.810</td>
</tr>
</tbody>
</table>
Figure 5. History of direction of thrust.

Table 8. Results of five trials using PSASM

<table>
<thead>
<tr>
<th>Solution</th>
<th>Run1</th>
<th>Run2</th>
<th>Run3</th>
<th>Run4</th>
<th>Run5</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\phi_1$</td>
<td>0.352042</td>
<td>-0.057372</td>
<td>0.259813</td>
<td>0.527462</td>
<td>0.348295</td>
</tr>
<tr>
<td>$\phi_2$</td>
<td>0.589395</td>
<td>1.20942289</td>
<td>0.253441</td>
<td>0.858197</td>
<td>1.304801</td>
</tr>
<tr>
<td>$\phi_3$</td>
<td>0.628842</td>
<td>-2.00602180</td>
<td>0.909478</td>
<td>1.732000</td>
<td>1.357893</td>
</tr>
<tr>
<td>$\phi_4$</td>
<td>1.458068</td>
<td>0.36343452</td>
<td>-0.845056</td>
<td>-0.206089</td>
<td>0.549036</td>
</tr>
<tr>
<td>$\phi_5$</td>
<td>-1.458497</td>
<td>-2.52802901</td>
<td>-0.835542</td>
<td>-1.384291</td>
<td>-0.271159</td>
</tr>
<tr>
<td>$\phi_6$</td>
<td>1.146520</td>
<td>0.73008010</td>
<td>-1.196995</td>
<td>0.744410</td>
<td>0.513495</td>
</tr>
<tr>
<td>$\phi_7$</td>
<td>-1.410176</td>
<td>-1.92264303</td>
<td>-1.851865</td>
<td>0.426516</td>
<td>-0.665769</td>
</tr>
<tr>
<td>$\phi_8$</td>
<td>0.686073</td>
<td>1.88674605</td>
<td>-1.559856</td>
<td>2.032576</td>
<td>1.364476</td>
</tr>
<tr>
<td>$\phi_9$</td>
<td>-1.333460</td>
<td>-0.82943904</td>
<td>-1.226154</td>
<td>-0.465361</td>
<td>-0.819916</td>
</tr>
<tr>
<td>$\phi_{10}$</td>
<td>-0.523628</td>
<td>-0.95342565</td>
<td>-1.121665</td>
<td>-0.698841</td>
<td>-0.452109</td>
</tr>
<tr>
<td>$T_w$</td>
<td>422</td>
<td>421</td>
<td>453</td>
<td>387</td>
<td>387</td>
</tr>
<tr>
<td>$m(t_f)$</td>
<td>3735.1</td>
<td>3735.1</td>
<td>3735.1</td>
<td>3870.0</td>
<td>3870.0</td>
</tr>
</tbody>
</table>
5. IMPULSIVE RENDEZVOUS OPTIMIZATION

The fuel-optimal rendezvous problem has been studied extensively, and many algorithms have been proposed [53-61]. The existing algorithms can guarantee obtaining a global solution for a linearized impulsive rendezvous, but are not as efficient and effective in solving nonlinear rendezvous problems. Widely used tools for optimizing a nonlinear impulsive rendezvous are the primer vector method and the Lambert algorithm [55-56]. Using these tools and classical gradient-based optimization algorithms, Jezewski and Rozendaal [56], Gross and Prussing [57], Prussing and Chiu [58], and Hughes et al. [60] successfully solved different types of unperturbed two-body multiple-impulse rendezvous problems. However, owing to limitations of gradient-based optimization methods, these approaches cannot guarantee a global solution, as was concluded by Hughes et al. [60] and was also observed in our experiments. Combining primer vector theory and the global convergence ability of PSASM, we propose a new interactive optimization approach for the fuel-optimal nonlinear rendezvous problem.

5.1. Multi-impulse Rendezvous Problem

For the general dynamic equations

\[
\begin{align*}
\dot{r} &= v \\
\dot{v} &= -\mu \frac{r}{r^3} + \Gamma 
\end{align*}
\]

(24)

where the thrust acceleration is approximated by impulses, i.e.,

\[\Gamma = \sum_{i=1}^{\infty} \Delta v_i \delta(t-t_i),\]

the initial and terminal conditions, respectively, for rendezvous are \( r_{0}, v_{0}, t_{0} \) and \( r_{f}, v_{f}, t_{f} \).

Assuming an impulse \( \Delta v_i \) is applied, with the superscript ‘−’ indicating the state before the impulse and ‘+’ indicating the state after the impulse, we get

Complimentary Contributor Copy
\[ \begin{align*}
    \mathbf{r}_i^+ &= \mathbf{r}_i^- \\
    t_i^+ &= t_i^- \\
    \Delta \mathbf{v}_j &= \mathbf{v}_j^+ - \mathbf{v}_j^-
\end{align*} \] (25)

Without loss of generality, let

\[ \begin{align*}
    \mathbf{r}_i &= \mathbf{r}_i^+ = \mathbf{r}_i^- \\
    t_i &= t_i^+ = t_i^-
\end{align*} \] (26)

and assume that \( \mathbf{r}(t + \Delta t) = f(\mathbf{r}(t), \mathbf{v}(t), t, t + \Delta t) \) and \( \mathbf{v}(t + \Delta t) = g(\mathbf{r}(t), \mathbf{v}(t), t, t + \Delta t) \) are the solutions to (24).

For an intermediate impulse \( i \neq 1, i \neq n \), where \( n \geq 2 \) is the number of impulses, the following conditions must be satisfied:

\[ \begin{align*}
    \mathbf{r}_i &= f(\mathbf{r}_{i-1}, \mathbf{v}_{i-1}^+, t_{i-1}, t_i) \\
    \mathbf{v}_i^- &= g(\mathbf{r}_{i-1}, \mathbf{v}_{i-1}^+, t_{i-1}, t_i)
\end{align*} \] (27)

The chosen independent variables, i.e., the optimization variables, are impulse times and the first \( n-2 \) impulse vectors

\[ \begin{align*}
    t_i &\quad i = 1, 2, \ldots, n \\
    \Delta \mathbf{v}_j &\quad j = 1, 2, \ldots, n - 2
\end{align*} \] (28)

and the objective function is

\[ J = \Delta \mathbf{v} = \sum_{i=1}^{n} |\Delta \mathbf{v}_i| \] (29)

Calculating \( \mathbf{r}_i \) and \( \mathbf{v}_i^- \), and \( \mathbf{r}_n \) and \( \mathbf{v}_n^+ \) using (27), we have

\[ \begin{align*}
    \mathbf{v}_i^+ &= \mathbf{v}_i^- + \Delta \mathbf{v}_i \\
    \mathbf{r}_{i+1}^+ &= f(\mathbf{r}_i, \mathbf{v}_i^+, t_i, t_{i+1}) \quad (i = 1, 2, \ldots, n - 2) \\
    \mathbf{v}_{i+1}^- &= g(\mathbf{r}_i, \mathbf{v}_i^+, t_i, t_{i+1})
\end{align*} \] (30)

Complimentary Contributor Copy
The rendezvous conditions are satisfied by solving a Lambert problem \( \text{Lambert}(r_{n-1}, r_n, t_n - t_{n-1}) \). With the solution for the Lambert problem, we can solve for \( \Delta v_{n-1} \) and \( \Delta v_n \) and then obtain the total \( \Delta v \).

5.2 Interactive Optimization Approach

The proposed interactive optimization approach can be outlined as follows.

\textit{Step 1 Two-impulse reference solution}

1.1 Let \( n = 2 \), and assume the initial and final impulses are applied at \( t_0 \) and \( t_f \), respectively. Obtain the two-impulse reference solution by using the Lambert algorithm.
1.2 Obtain the primer vector history by using the shooting algorithm, and determine whether it satisfies Lawden’s conditions.
1.3 If so, go to Step 1.4; otherwise, go to Step 2.1.
1.4 Optimize the initial and final impulse times. If a smaller \( \Delta v \) is obtained, go to Step 2.3; otherwise, the two-impulse reference solution is the optimal solution, end.

\textit{Step 2 Add initial coast or final coast}

2.1 Determine whether to add initial and final coast by observing the primer magnitude figure. For a two-impulse reference trajectory, Figure 6 shows four representative primer vector histories corresponding to different improvement methods.
2.2 If the primer magnitude figure matches Figure 6c or 6d, the initial or final coast should be added. With the initial and final impulse times as optimization variables, obtain the improved two-impulse solution.
2.3 Determine whether this improved solution satisfies Lawden’s conditions. If so, end; otherwise, go to Step 3.

\textit{Step 3 Search mid-impulse}

3.1 Add a mid-impulse: \( n = n + 1 \).
3.2 Execute the following optimization iteration: use evolutionary algorithms to obtain the \( n \)-impulse optimal solution using the multi-impulse Lambert optimization programming models, as described in section 3.2.

\textit{Step 4 Re-compute the n-impulse solution}
4.1 Obtain the primer magnitude history corresponding to this \( n \)-impulse and determine whether this \( n \)-impulse solution satisfies Lawden’s conditions.

4.2 If so, end. Otherwise, considering the stochastic character of the evolutionary algorithms, re-execute Step 3.2 ten times and determine whether the best solution of these ten runs satisfies Lawden’s conditions.

4.3 If so, end; otherwise, go to Step 3.1.

Figure 6. Four representative primer magnitude categories. a) Optimal. b) Non-optimal (additional impulse required). c) Non-optimal (initial coast required). d) Non-optimal (final coast required).

In this optimization approach, the improvements, including adding a mid-impulse, are determined by observing the primer magnitude histories. Thus, the approach requires the designers’ participation and is interactive. Figure 6 provides only four representative primer magnitude histories. In fact, primer vector histories in practical application are not limited to these. Practical design would require more designer experience.
5.3. Numerical Optimization Results

The proposed approach has been tested for three cases of rendezvous: circle-to-circle, same-circle, and noncoplanar [62]. Herein, the results of the third test case, a more practical rendezvous problem, are provided. We describe the rendezvous problem using the classical orbit elements $E = (a, i, e, \Omega, \omega, \nu)$, where $a$ is the semimajor axis, $i$ is the inclination, $e$ is the eccentricity, $\Omega$ is the right ascension of the ascending node (RAAN), $\omega$ is the argument of perigee, and $\nu$ is the true anomaly. The rendezvous initial conditions are:

- $E_{tar} = (6771.1 \text{ km}, 42^\circ, 0, 120^\circ, 0, 180^\circ)$
- $E_{cha} = (6741.1 \text{ km}, 42.1^\circ, 0, 120.2^\circ, 0, 175^\circ)$

The full transfer time equals two target orbital periods, $t_f = 11107.2 \text{ s}$.

Table 9. Solutions for the noncoplanar rendezvous problem

<table>
<thead>
<tr>
<th>Index</th>
<th>Impulses ($t_i (s)$, $\Delta v_i (m/s)$)</th>
<th>$\Delta v$ (m/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>i=1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Two-impulse reference</td>
<td>(0, 11740.9)</td>
<td>(11107.2, 11708.6)</td>
</tr>
<tr>
<td>Two-impulse best</td>
<td>(6643.7, 37.3)</td>
<td>(10689.4, 16.2)</td>
</tr>
<tr>
<td>Three-impulse best</td>
<td>(0, 7.9)</td>
<td>(6373.4, 18.0)</td>
</tr>
<tr>
<td>Four-impulse optimal</td>
<td>(593, 5.6)</td>
<td>(6809.0, 10.2)</td>
</tr>
</tbody>
</table>

Figure 7a plots the primer-magnitude time history of the two-impulse reference solution, which seems to satisfy Lawden’s conditions. However, as the $\Delta v$ is 23,449.5 m/s, it should not be the fuel-optimal solution. The $\Delta v$ is greatly reduced to 53.5 m/s by optimizing the initial and final impulse times. The obtained two-impulse best solution’s primer-magnitude time history is plotted in Figure 7b, the latter part resembling Figure 6b. Therefore, a mid-impulse should be added. The three-impulse solution’s primer-magnitude time history is plotted in Figure 7c, which still does not satisfy Lawden’s
conditions. Another mid-impulse is added, and the four-impulse solution is obtained; its primer-magnitude time history is plotted in Figure 7d, which satisfies Lawden’s conditions. These four solutions for this noncoplanar rendezvous problem are listed in Table 9.

Figure 7. (Continued).

(a) Two-impulse reference solution  
(b) Two-impulse best solution
In the gradient-based optimization approach in [56-58, 60], the optimization process is greatly dependent on the primer vector. However, this type of approach is subject to convergence problems in obtaining global solutions for complex rendezvous problems.
These problems can be overcome in our proposed interactive approach. Only the primer-magnitude history is observed to define whether to add a mid-impulse, and no further information related to the primer vector is used. The mid-impulses are searched directly in a large-scale space by using PSASM to solve an $n$-impulse problem. As PSASM has good global convergence ability, it can easily determine the best $n$-impulse solution. Table 10 provides the PSASM statistical results in 50 independent runs for each configuration of the noncoplanar rendezvous problem. Figure 8 plots the PSASM convergence time histories. From Table 10 and Figure 8, we can conclude that PSASM is efficient and effective in obtaining the best solution for the $n$-impulse problem. If the $n$-impulse solution is found to be better than the $(n-1)$-impulse solution and also satisfies Lawden’s conditions, we can be confident that the $n$-impulse is the final global solution. For the noncoplanar rendezvous problem, the four-impulse solution is better than the three-impulse one and also satisfies Lawden’s conditions. In order to verify the optimality of the four-impulse solution, we further test the five-impulse case. Figure 9 plots the primer magnitude of the five-impulse best solution, and Table 10 reports the statistical results. From Table 10, the $\Delta v$ of the five-impulse is larger than the four-impulse. Besides, as demonstrated in Figure 9, the second-impulse position always converges to overlap the first-impulse position. This also confirms that the four-impulse solution is the optimal solution.

<table>
<thead>
<tr>
<th>Index</th>
<th>$\Delta v$ (m/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two-impulse best</td>
<td>53.5140 53.5140 53.5140 0</td>
</tr>
<tr>
<td>Four-impulse optimal</td>
<td>36.1974 36.2478 36.2122 0.0124</td>
</tr>
<tr>
<td>Five-impulse best</td>
<td>36.2251 38.7801 36.7402 0.6409</td>
</tr>
</tbody>
</table>

Table 10. Statistical results of PSASM for the noncoplanar rendezvous (50 independent runs for each configuration)
CONCLUSION

This chapter presented a new kind of parallel simulated annealing using the simplex method (PSASM) that combines the advantages of simulated annealing with that of the simplex method. Experimental results were presented for several example mathematical problems. Using only a very simple penalty function, PSASM obtained the best known solutions for two benchmark nonlinear constrained optimization problems, including Golinski’s speed reducer problem, with the lowest computational cost.

PSASM was applied to two representative types of spacecraft trajectory optimization problem. The first problem was low-thrust trajectory optimization. The minimum-transfer-time and minimum-fuel-consumption optimizations of a low-thrust Earth-to-Mars orbit transfer were illustrated using PSASM. The results demonstrated that PSASM is superior to SA, GA,
and the classical optimization algorithms. Master/client parallel computation is implemented using MPI, and the experiment confirmed the theoretical result. PSASM demonstrated an excellent parallel efficiency.

The second problem was impulsive rendezvous trajectory optimization. A new optimization approach combining primer vector theory and PSASM for a fuel-optimal non-linear impulsive rendezvous was proposed. The optimization approach was designed to seek the optimal number of impulses as well as the optimal impulse vectors. In this optimization approach, PSASM was used to optimize the rendezvous trajectory with a fixed-number of impulses. The results showed that the optimization approach is effective and efficient in fuel-optimal non-linear rendezvous design. PSASM was demonstrated to be able to guarantee solutions that satisfy Lawden’s necessary optimality conditions.

Figure 9. The primer-magnitude time history of the five-impulse best solution for the noncoplanar rendezvous ($\Delta v = 36.22$ m/s).

**REFERENCES**


Complimentary Contributor Copy


Complimentary Contributor Copy


Chapter 6

SIMULATED ANNEALING APPLICATIONS IN THERMODYNAMIC CALCULATIONS

Ahmed O. Elnabawy¹, Seif-Eddeen K. Fateen¹ and Adrian Bonilla-Petriciolet²

¹Cairo University, Department of Chemical Engineering, Cairo, Egypt
²Instituto Tecnológico de Aguascalientes, Department of Chemical Engineering, Aguascalientes, México

ABSTRACT

Applied thermodynamic calculations are relevant for process design. These thermodynamic calculations include: phase equilibrium calculations, chemical equilibrium calculations, simultaneous phase and chemical equilibrium calculations, phase stability analysis, calculation of critical points, calculation of azeotropes, and parameter estimation of thermodynamic models, among others.

These thermodynamic calculations can be solved as global optimization problems. In particular, Simulated Annealing (AS) has shown a promising performance in solving these thermodynamic problems. This manuscript overviews the application of AS and its variants, including hybrid method, in applied thermodynamic calculations for process design.

Keywords: Simulated annealing, global optimization, thermodynamic calculations

Complimentary Contributor Copy
1. INTRODUCTION

Optimization means finding the best solution as judged by a tailored objective function. The objective function needs be minimized or maximized in order to obtain the best set of values of its decision variables that satisfy prescribed constraints. Mathematically, this can be defined as:

\[
\text{Min } F(X)
\]

with respect to \( D \) decision variables \( X = (X^1, \ldots, X^D) \). The upper and lower bounds of these variables are: \( X^i_{\text{min}} \leq X^i \leq X^i_{\text{max}} \). Two types of minima are known: local and global minima. The local minimum is defined to be the minimum value of the objective function within a confined neighborhood. On the other hand, the global minimum is the minimum value of the objective function throughout its whole feasible domain. It is clear that finding the global optimum is more challenging than finding the local optima. To find a global optimum, one has to be cautious not to be trapped in a local one, and if an entrapment may occur, procedures to free the solution from the local minimum zones should be employed. It is, of course, the global minimum that is sought by chemical engineers to optimize their calculations, being the best solution. In applied thermodynamic calculations, however, the global minimum often constitutes the only correct solution [1]. Applied thermodynamic calculations include, but are not restricted to: phase equilibrium calculations, chemical equilibrium calculations, simultaneous phase and chemical equilibrium calculations, phase stability analysis, calculation of critical points, and parameter estimation of thermodynamic models.

Specifically, phase and chemical equilibrium calculations are usually performed via the minimization of the Gibbs free energy of the system. The expression for Gibbs free energy could include various models for the phases at hand. The models could be ideal or non-ideal. On the other hand, phase stability analysis is usually carried out via the global minimization of the tangent plane distance function. Also, the parameter estimation in thermodynamic models is usually based on comparing the phase behavior of a system as obtained by the model to a set of experimental data. In this case, the objective function should represent this comparison in an appropriate manner. The use of local minima in such thermodynamic calculations could result in erroneous phase calculations, including inability to recognize phase splits.
whether an azeotrope is homogeneous or heterogeneous, to name a few [2]. All these situations could lead to uncertainty in process design.

The objective functions involved in applied thermodynamic calculations are usually non-linear, and potentially non-convex with several local minima within the specified bounds, which further complicates the task of finding the global minimum [1]. Global optimization methods have been used for applied thermodynamic calculations and they are usually divided into: deterministic and stochastic. Deterministic methods search systematically for the minimum within the prescribed range of the decision variables, while stochastic methods employ a random search and heuristics to find the global minimum [2].

Although deterministic methods offer theoretical and computational guarantees to find the global minimum, stochastic methods retain several advantages. They do not require any assumptions for the problem at hand, are capable of addressing the non-linearity and non-convexity of the objective function as a black box problem, and are relatively easier to program and implement [1]. Moreover, these methods have their extrapolations to multi-objective optimization, and are proper to parallelization to efficiently handle large scale problems [3]. Also, from the perspective of practical applications, they reduce significantly the optimization time as compared to deterministic methods, especially for large scale problems.

As stated in previous chapters, Simulated Annealing (AS) is a stochastic global optimization technique (SGO), which has been found useful in providing feasible solutions to a variety of practical and real-life optimization problems in different science and engineering fields including thermodynamic calculations [4]. AS has shown great promise in solving thermodynamic calculations. In this chapter, we provide a general overview of the application of AS for solving applied thermodynamic calculations. Hence, the remainder of this work is organized as follows: section 2 discusses AS and its variants and some hybrid algorithms, section 3 details the different thermodynamic calculations reviewed, together with outlining a plethora of studies that employ AS in thermodynamic applications. Finally, a conclusion is provided in section 4.

2. SIMULATED ANNEALING

Simulated annealing mimics the thermodynamic process of slowly cooling a molten metal to crystallize. It was originally suggested by Kirkpatrick et al. [5] recognizing an analogy between statistical mechanics and thermodynamics.
of annealing processes. AS is a point-to-point random search algorithm, which drives an objective function (or by analogy, energy). At a certain temperature (pseudo-temperature, which is an inherent parameter of this algorithm), a random point is generated, and the energy of the system (i.e., the objective function) is evaluated. The energy just calculated is compared to that calculated for the previous point (generated randomly as well); if the new calculated energy is smaller the point is accepted, if the new calculated energy is higher the point is accepted with a certain probability that usually follows Boltzmann rule, in what is known as the Metropolis criterion. The latter feature simulates the premature formation of an imperfect crystal (not at its lowest energy), and it is this feature that helps AS algorithms escape from local minima. This probability decreases steadily as the temperature moves down via a prescribed cooling scheme, after a number of iterations has driven the system to “quasi-equilibrium” at the current temperature [1, 2, 6]. The probability of accepting moves with higher cost function values, which decreases during the course of optimization, is a central feature of threshold algorithms to which, therefore, AS belongs. The cooling schedule could even vary during the run to better meet the varying needs of intensification and diversification at different stages of optimization. For example, reheating (raising the value of the temperature parameter) would raise back the probability of accepting uphill moves, favoring diversification at the expense of intensification [7].

Cooling schedules are generally divided into static (where schedule parameters are fixed), and dynamic schedules (in which parameters vary and adapt during the algorithm run) [8-10]. A brief review of AS, its convergence analysis, and its performance in applications is given by [11, 12]. Ingber [13] provides an enlightening discussion of AS, ASA, and simulated quenching (SQ), as well as a variety of applications. A study of AS variants is provided in [14].

This stochastic algorithm is often used in conjunction with a local optimizer called “the Nelder and Mead simplex algorithm”. The initial temperature at which the algorithm starts is calculated via heuristic rules. The simplex is then started to calculate a local minimum at this temperature. The simplex is then perturbed to allow for uphill or downhill movements (for minimization or maximization respectively), and then it searches for another minimum and so on.

A certain equilibrium criterion is applied to avoid prohibitive computing time. When the equilibrium criterion is met, the whole algorithm is repeated at a lower annealing temperature obtained via the cooling scheme. A stopping
criterion is usually employed; typically, the minimum annealing temperature a cooling scheme could finally reach [1, 2].

Improvements on AS have focused on combining it with local search methods, making AS adaptive, altering the neighborhood structure, and altering the acceptance criterion and hill climbing techniques, among other improvements. We provide a brief overview of alternatives to improve the performance of AS:

- The acceptance criterion of AS was combined to chained local optimization (CLO) where, starting from a local minimum, a perturbation takes place, then the new point is minimized using a local optimizer, and then the acceptance criterion of AS is employed to either accept or reject the minimized perturbed trial. The authors have also run CLO in parallel, in what was deemed parallel CLO, which shared certain aspects of Genetic Algorithm namely, the presence of an interacting population of candidate solutions [15].

- Annealing Adaptive Search, or AAS, has been applied to discrete and mixed continuous/discrete optimization problems. AAS is an ideal form of AS in which a number of points is generated following a Boltzmann’s distribution. Monte Carlo method with Markov chains have been used to sample the Boltzmann distribution generated points, of which the special Hit-and-Run method is used as sampler. The authors developed an analytical cooling schedule tailored for AAS in order for it to reach global minimum [16].

- Thermodynamic AS, or TSA, has been proposed as a means of making the tedious experiments of tuning schedule parameters. TSA makes use of internal energy and entropy state variations for guiding the temperature parameter through its evolution. Based on the first and second laws of thermodynamics, the temperature is being calculated, instead of being determined initially or according to a prescribed schedule. Being less sensitive to initial solution and initial temperature, TSA lends itself easily to optimization in different stages and optimization from pre-optimized points [17].

- AS has been used in conjunction with multiple neighborhood structures, and with an optimal stopping criterion in which the temperature is changed at the right time. The $K$ neighborhood structures are often problem-specific, and one of them is selected for solution. The optimal stopping criterion is then employed to determine whether to stop searching this specific neighborhood or
continue. If stopped, the search might resort to another neighborhood at the same temperature, or reduce the temperature and repeat the procedure all over again [18].

- AS has been used for locating multiple global minima. The objective function is being transformed iteratively by means of a function stretching technique, and the objective function is penalized to convert the constrained problem to unconstrained one by penalizing the objective function if any of the constraints is violated. A comparative study with three penalty functions has been conducted in [19]. Similar work was conducted using the ASA variant [20]. Multiple global maxima have been detected as well using AS with a function stretching technique [21].

- Pedamallu et al. [22] have suggested a hybrid AS, or HSA, for the solution of constrained optimization problems. HSA does not accept a hill-climbing move except after a specified number of trials has failed to discover a decreasing or an equal cost solution, and it is coupled with the local algorithm Feasible Sequential Quadratic Programming. Two versions of HSA were presented, namely: HSA with penalty terms, or HSAP, to deal with constraint violations by infeasible solutions, and HSAD in which HSA is applied with dual sequence for feasible and infeasible solutions separately. Numerical tests have been used to judge the performance of the two versions, and to compare them to other SA variants.

Other variants of AS include parallel AS [23, 24] and its hybridization with other metaheuristics (e.g., Genetic Algorithms, Particle Swarm Optimization, etc.); while other applications of AS include: parameter estimation and feature selection in support vector machine [25].

3. APPLICATIONS OF SIMULATED ANNEALING IN APPLIED THERMODYNAMIC CALCULATIONS

In this section, we provide a brief description of relevant thermodynamic calculations for chemical engineering applications using AS.

It is convenient to remark that we have used the problem formulation and description reported by Zhang et al. [4] for the selected thermodynamic problems.
3.1. Phase Stability

Solving the phase stability (PS) problem is usually the starting point for the phase equilibrium calculations. The theory used to solve this problem states that a phase is stable if the tangent plane generated at the feed (or initial) composition lies below the molar Gibbs energy surface for all compositions. The traditional implementation of the tangent plane criterion \([26-28]\) is to minimize the tangent plane distance function (TPDF), defined as the vertical distance between the molar Gibbs energy surface and the tangent plane at the given phase composition.

Probably, the first application of AS in applied thermodynamic calculations for process design was the phase stability analysis. Specifically, Rangaiah \([29]\) compared AS and Genetic Algorithm in phase equilibrium and phase stability calculations for a variety of systems including binary, ternary, quaternary, and nine-hydrocarbon systems. AS was found to be robust in TPDF minimization, but it fails to globally minimize the Gibbs free energy for some complex systems. In this study, both algorithms GA and AS were hybridized and showed an enhanced performance for solving complex problems. On the other hand, Bonilla-Petriciolet et al. \([30]\) used different versions of AS namely: conventional AS, very fast AS, a modified version of direct search AS, and stochastic differential equations to perform phase stability analysis for a variety of reactive and non-reactive multi-component mixtures. They concluded that AS is the best algorithm, while the very fast AS is the worst, and usually converges to local minimum. Zhu et al. \([31]\) have reported the use of an enhanced AS algorithm for stability analysis for high-pressure systems described by SRK or PR EoS, while Pereira et al. \([32]\) have performed stability analysis to a ternary and a quaternary system using NRTL model. The minimization strategy was the Stretched AS (SSA). It couples Adaptive AS (ASA) with a function stretching mechanism. ASA uses a probability density function to generate a new starting point for the next generation. It also considers re-annealing; that is, redefining the control (or cooling schedule) parameters during iterations. Stretching technique was originally made to help particle swarm algorithm escape from local minima. It depends on a two-stage transformation of the objective function when a local minimum is found. SSA was found to give successful results, as judged by comparison with literature data, for the minimization of the TPDF, especially when coupled with a local minimizer.

Souza et al. \([33]\) have reported the modeling of experimental data for the systems: \(\text{CO}_2 + \text{clove oil}\), while Moura et al. \([34]\) have reported experimental
data for the systems CO$_2$ + fennel extract [34]. These systems were modeled as pseudo-binary using the PR EoS with quadratic mixing rules. Here, the authors recognized the need to phase stability analysis before performing phase split calculations. Hence, phase stability analysis was performed by minimizing the tangent plane distance of the Helmholtz free energy via the interval Newton/generalized bisection technique in [33], and by minimizing the tangent plane distance of the Gibbs free energy via AS in [33, 34]. In [33], both minimization strategies gave the same results; however, the stochastic algorithm was more efficient in terms of computational time. Finally, Nagatani et al. [35] have performed phase stability analysis for a number of LLE systems modeled using NRTL model. Two approaches for solving the phase stability problem were explored: solution of a number of equations via a subdivision algorithm, and minimization of TPDF using AS. In general, both approaches were able to identify the global minima, and the CPU time was lower for the subdivision algorithm in the binary and ternary systems, but increased dramatically in the quaternary and higher order systems. Hence, AS is preferable for phase stability of multicomponent systems.

3.2. Phase Equilibrium Calculations

A mixture of substances at a given temperature, $T$, pressure, $P$ and total molar amount may separate into two or more phases. The composition of the different substances is the same throughout a phase but may differ significantly in different phases at equilibrium. If there is no reaction between the different substances, then it is a phase equilibrium problem. There are mainly two approaches for PEC: equation solving approach and Gibbs free energy minimization approach. The former involves solving a set of non-linear equations arising from mass balances and equilibrium relationships. The latter involves the minimization of the Gibbs free energy function. Although the first approach seems to be faster and simple, the solution obtained may not correspond to the global minimum of Gibbs free energy function. Moreover, it needs a priori knowledge of phases existing at equilibrium [29]. Classic thermodynamics indicate that minimization of Gibbs free energy is a natural approach for calculating the equilibrium state of a mixture. Hence, the Gibbs free energy minimization is commonly used to determine phase compositions at equilibrium [29, 36-38].

The mathematical formulation of PEC involves the minimization of Gibbs free energy subject to mass balance equality constraints and bounds that limit

Complimentary Contributor Copy
the range of decision variables. Different AS algorithms have been used in several studies for performing the Gibbs free energy minimization. Specifically, Bonilla-Petriciolet et al. [41, 42] have assessed DSAM (a special version of AS algorithm) for Gibbs energy minimization for flash calculations. They reported very encouraging results with a high success rate but at the expense of a significant CPU time. In other study, Henderson et al. [43] presented a new formulation to perform flash calculations, which was formulated as an optimization problem and was satisfactorily solved using AS. Corazza et al. [44] have used AS to perform SVL equilibrium calculations; Pan and Firoozabadi [45] have used AS to directly minimize Gibbs free energy for several multicomponent, multiphase systems spanning: VLE, VLLE, and VLSE. They showed that AS works well in all systems studied and does not require special treatment for critical calculations.

3.3. Phase and Chemical Equilibrium Calculations

In phase and chemical equilibrium (PCE) problems, also known as reactive phase equilibrium problems, reactions increase the complexity and dimensionality of phase equilibrium problems, and so phase split calculations in reactive systems are more challenging due to non-linear interactions among phases and reactions. The phase distribution and composition at equilibrium of a reactive mixture are determined by the global minimization of Gibbs free energy with respect to mole numbers of components in each phase subject to element/mass balances and chemical equilibrium constraints [46, 47,48]. The expressions for Gibbs free energy and its mathematical properties depend on the structure of the thermodynamic equation(s) chosen to model each of the phases that may exist at equilibrium. The application of AS for PCE problems involves both constrained and unconstrained problem optimization problems. For example, Reynolds et al. [37] have used the AS in chemical and phase equilibrium via the minimization of Gibbs free energy. On the other, Bonilla-Petriciolet et al. [49] used AS in the global minimization of Gibbs free energy for two-phase equilibrium calculations in reactive mixtures. Recently, Bonilla et al. [48] have compared AS, GA and differential evolution with tabu list (DETL) in the context of minimizing Gibbs free energy in the problem of VLE and LLE in reactive mixtures. They formulated the problem as both constrained and unconstrained optimization using both conventional and transformed composition variables. They reported that both DETL and AS are
superior to GA in terms of reliability in the context of constrained and unconstrained Gibbs energy minimization.

3.4. Parameter Estimation and Thermodynamic Modeling

The estimation of parameters in thermodynamic models is an important requirement and a common task in many areas of chemical engineering because these models form the basis for synthesis, design, optimization and control of process systems. In the case of separation processes, thermodynamic models play a major role with respect to energy requirements, phase equilibrium and equipment sizing. The parameter estimation problem refers to determining values of model parameters that provide the best fit to a set of measured data such as vapor-liquid or liquid-liquid equilibrium. In particular, estimation of parameters in non-linear thermodynamic models for vapor-liquid equilibrium (VLE) modeling has been of great interest in the chemical engineering literature. VLE data modeling using thermodynamic equations is generally based on classical least squares or maximum likelihood approaches [51]. In the classical least squares, it is assumed that there is a set of independent variables not subject to measurement error and only the dependent variables have errors, while errors in all measured variables are accounted in the maximum likelihood approach. AS has been widely used in parameter estimation of thermodynamic models. For example, Behazdi et al. [2] have employed the simplex AS algorithm for the problem of optimizing the parameters for the SAFT-VR EoS. The results of the stochastic algorithm are found superior as compared to those found by a gradient-based quasi-Newton method in six out of seven pure solvents, and in nine out of eleven electrolytes. In other study [52], the NRTL model parameters were correlated to homogeneous azeotropic mixtures of water + 1,2 ethanediol and 2,3-dimethyl-2-butane + methanol using AS and PSO. The objective function for optimization was written as to compare the optimized parameters to experimental data found in DECHEMA. These authors concluded that both stochastic solvers are reliable, although PSO usually performed poorer than AS. Bonilla et al. [53] obtained the parameters of NRTL model using AS, and applied those during the optimization of homogeneous azeotropic distillation columns. Results of this study suggested significant differences in optimization variables and operation and control schemes. Also, Bonilla-Petriciolet et al. [54] have evaluated AS for parameter estimation of different VLE systems using different thermodynamic models. Bonilla-Petriciolet et al.
[41, 42] have studied the performance of DSAM (a special version of AS algorithm) for data fitting in thermodynamic models for electrolytes. They reported unsatisfactory results with a success rate of only 64%, and hence reinforced that DSAM is not suitable for electrolyte parameter estimation. Ferrari et al. [55] have employed AS and PSO in parameter estimation of NRTL and UNIQUAC models in LLE calculations coupled with phase stability test. Both optimization strategies showed a good performance for determining the model parameter values, but with AS requiring less computational effort than PSO for binary and ternary systems, and PSO requiring less computational effort than AS for quaternary systems.

Kundu et al. [56, 57] have demonstrated the supremacy of AS as compared to the traditional deterministic Levenberg-Marquardt, or LM, technique in the parameter estimation of the modified Clegg-Pitzer equation to correlate and predict VLE of: CO$_2$ in aqueous solutions of 2-amino-2-methyl-1-propanol [56] and CO$_2$ in aqueous N-Methyldiethanolamine [57]. Five interaction parameters were fit to minimize the error of CO$_2$ partial pressure in [56], while three interaction parameters were fit to minimize the squared error of CO$_2$ partial pressure in [57]. In later studies, Kundu et al. [58] and Anil et al. [59] have identified the supremacy of DE to AS and LM when solving the same type of parameter estimation problems. Other studies have reported the application of AS for modeling the solubility of CO$_2$ in water + diethanolamine (DEA) + 2-amino-2-methyl-1-propanol (AMP) [60]; the solubility of CO$_2$ in water + diethanolamine (DEA) + N-methyldiethanolamine (MDEA) [61], the experimental VLE and SFE data for binary mixtures of progesterone in compressed and/or supercritical CO$_2$, propane, and n-butane using the PR EoS [63], and the VLE, along with heat of mixing and activity coefficients for various aqueous alkanolamine solutions using the NRTL model [64].

3.5. Critical Points and Azeotropes

Both critical points and azeotropes are special cases of phase equilibrium calculations, and are important topics because of their theoretical and practical implications for design of separation processes. They can occur in both reactive and nonreactive mixtures, and a mixture may have one, more than one or no critical points and/or azeotropes. Therefore, the correct prediction of phase behaviour requires the determination of all critical points and

Complimentary Contributor Copy
azeotropes, or determining with certainty that there are none in the domain of interest [65, 66].

For nonreactive mixtures, homogeneous azeotropes occur when the compositions of vapour and liquid phases at equilibrium are identical. The same definition applies for homogeneous azeotropes in reactive systems but using reaction-invariant composition space [67]. For a multicomponent mixture, the thermodynamic conditions that a homogeneous azeotrope should satisfy are based on the equality of chemical potentials [4]. On the other hand, the definition of the critical point of a mixture is as follows: at the critical point, the intensive properties of two phases in equilibrium become identical [68]. In particular, the Heidemann-Khalil formulation of the criticality conditions is the most widely used criteria in the literature for calculation of critical points [69]. All of these problem formulations can be used for prediction of both homogeneous azeotropes and critical points in reactive systems if reaction invariant composition variables are used [67, 70]. Note that the prediction of both azeotropes and critical points is reduced to locating all roots of a system of non-linear equations [4]. Such systems of equations can be transformed to an optimization problem in order to use global optimization methods. Overall, AS has been one of the few stochastic optimization methods used for predicting azeotropes and critical points. For example, Bonilla-Petriciolet et al. [41, 42] have assessed DSAM (a special version of AS algorithm) for calculation of homogeneous azeotropes, and they reported very encouraging results with 100% reliability for both reactive and non-reactive systems.

Later, Sánchez-Mares and Bonilla-Petriciolet [71] have formulated the problem of finding critical points in binary and multicomponent mixtures as a global optimization problem. This approach was based on the Heidemann and Khalil criticality criterion, and several critical points could be located. AS was the minimization strategy. It was found that AS is a robust minimizer, but is expensive computationally, especially for systems with large number of components. Sanchez-Mares et al. [72] reported the critical point calculations for reacting mixtures via AS. They used the criterion of Heidemann and Khalil to calculate critical points, and coupled it with the framework of Ung and Doherty to reduce the problem dimensionality associated with chemical reactions. Justo-Garcia et al. [73] have employed AS to solve the criticality conditions required for identification of critical points in binary and multicomponent systems. The criticality conditions were based on the TPDF as applied to Helmholtz energy, with temperature and volume as independent
variables. SRK and PR EoS were used to model the phase behavior of the studied systems.

Henderson et al. [74] have modified the Gibbs criticality criterion for the identification of multiple critical points using AS. Finally, AS has also been used for the calculations of homogeneous azeotropes in reactive and non-reactive mixtures [67, 73].

4. CONCLUDING REMARKS AND DISCUSSION

Applied thermodynamic calculations have benefited significantly from the research in the field of stochastic global optimization in the last decade. In particular, Simulated Annealing is a stochastic point-to-point calculation, which has numerous successful applications in thermodynamic calculations. AS has parameters that have to be tuned to locate the global minimum reliably and efficiently.

The optimum values of these parameters represent the trade-off between reliability (finding the global minimum with close precision) and efficiency (as measured by computational effort). Hence, these parameters should provide for the subtle balance between intensification (exploitation) and diversification (exploration).

The problem of locating the subtle balance between intensification and diversification is very important. Currently, setting the parameters responsible for this task is conducted by means of tedious parametric studies, or by simply extracting their values from analogous previous studies in literature. The first procedure is very time consuming, and the second is prone to sub-optimal performance.

The big picture is to make AS amenable to be used widely in process simulators, with incurring minimum burden on the user. Thermodynamic calculations are solved thousands of times during process simulation, with this figure increasing manifolds for dynamic simulation. Being more effective in optimization, AS would be very useful for implementation in process simulators, only if their computation time is within practical bounds, and their use does not add extra burden on behalf of the user; a burden that is represented by extra parameters and variables that need be fixed before a simulation can run. This makes apparent the importance of this field of study, and the continuous need of improving existent AS, inventing new AS, and hybridizing them for improving the results obtained for thermodynamic calculations.
REFERENCES


Complimentary Contributor Copy
Simulated Annealing Applications in Thermodynamic Calculations


Complimentary Contributor Copy


Complimentary Contributor Copy
Simulated Annealing Applications in Thermodynamic Calculations


Complimentary Contributor Copy


SA OPTIMIZATION OF TWO-CHAMBER MUFFLERS EQUIPPED WITH MULTIPLE PARALLEL PERFORATED TUBES WITHIN A CONSTRAINED SPACE

Min-Chie Chiu*  
Department of Mechanical and Automation Engineering, Chung Chou University of Science and Technology, Taiwan, R.O.C.

ABSTRACT

Recently, research on mufflers hybridized with a single perforated tube has been addressed; however, the research work on mufflers conjugated with multiple parallel perforated tubes which may disperse the venting fluid and reduce secondary flowing noise has been neglected. Therefore, the main purpose of this paper is not only to analyze the sound transmission loss (STL) of two-chamber mufflers equipped with multiple parallel perforated tubes but also to optimize the best design shape within a limited space. In this paper, the four-pole system matrix for evaluating acoustic performance — sound transmission loss (STL) — is derived by using a decoupled numerical method. Moreover, a simulated annealing

* Corresponding author: Min-Chie Chiu. E-mail: minchie.chiu@msa.hinet.net. Mailing address: Department of Mechanical and Automation Engineering, Chung Chou University of Science and Technology, 6, Lane 2, Sec.3, Shanchiao Rd., Yuanlin, Changhua 51003, Taiwan, R.O.C.
(SA) method, a robust scheme used to search for the global optimum by imitating a physical annealing process, has been used during the optimization process. Before dealing with a broadband noise, the STL’s maximization with respect to a one-tone noise (300Hz) is introduced for a reliability check on the SA method. Additionally, an accuracy check of the mathematical model is performed. To appreciate the acoustical ability of two-chamber mufflers equipped with multiple parallel perforated tubes, three kinds of two-chamber mufflers hybridized with one, two, and four perforated tubes have been assessed and compared. Results reveal that the maximal STL is precisely located at the desired tone. In addition, the acoustical performance of two-chamber mufflers conjugated with more perforated tubes will decrease due to the decrement of the acoustical function for acoustical elements (II) and (III).

Consequently, to design a more efficient muffler, the maximum (allowable) flowing velocity within the muffler must be determined in advance before choosing the appropriate number of perforated tubes and performing the muffler’s shape optimization.

**Keywords:** Multiple parallel perforated tube; two-chamber; simulated annealing

**NOMENCLATURE**

This paper is constructed on the basis of the following notations:

- $c_o$: sound speed (m s$^{-1}$)
- $dH$: the diameter of a perforated hole on the inner perforated tube (m)
- $D_i$: diameter of the $i$th perforated tubes (m)
- $D_o$: diameter of the outer tube (m)
- $f$: cyclic frequency (Hz)
- $j$: imaginary unit
- $k$: wave number ($=\frac{\omega}{c_o}$)
- $kk$: cooling rate
- $f_1, f_2, f_3, f_4$: coefficients in function $f_i e^{\beta x}$
- $iter$: maximum iteration
- $L_{C1}, L_{C2}$: lengths of perforate straight ducts (m)
\( L_i \)  
length of the ith perforated tubes (m)

\( L_o \)  
total length of the muffler (m)

\( M_i \)  
mean flow Mach number at the \( i^{th} \) node

\( \text{OBJ} \)  
objective function (dB)

\( P_i \)  
acoustic pressure at the ith node (Pa)

\( pb(T) \)  
transition probability

\( Q \)  
volume flow rate of venting gas (m\(^3\) s\(^{-1}\))

\( \text{RT1} \)  
design parameter (RT1=D2)

\( \text{RT2} \)  
design parameter (RT2=D3/D2)

\( \text{RT3} \)  
design parameter (RT3=D4)

\( \text{RT4} \)  
design parameter (RT4=L3)

\( \text{RT5} \)  
design parameter (RT5=dH)

\( \text{RT6} \)  
design parameter (RT6=\( \sigma \))

\( S_i \)  
section area at the ith node (m\(^2\))

\( \text{STL} \)  
sound transmission loss (dB)

\( \text{SWLO} \)  
unsilenced sound power level inside the muffler’s inlet (dB)

\( \text{SWL}_T \)  
overall sound power level inside the muffler’s output (dB)

\( t_i \)  
the thickness of the ith inner perforated tube (m)

\( \text{TS1}_{ij}, \text{TS2}_{ij}, \text{TS3}_{ij}, \text{TS4}_{ij} \)  
components of four-pole transfer matrices for an acoustical mechanism with straight ducts

\( \text{TP1}_{ij}, \text{TP2}_{ij}, \text{TP3}_{ij}, \text{TP4}_{ij} \)  
components of a four-pole transfer matrix for an acoustical mechanism with an expanded perforated intruding tube

\( \text{TSC1}_{ij}, \text{TSC2}_{ij} \)  
components of a four-pole transfer matrix for an acoustical mechanism with a sudden contracted tube

\( \text{TSE1}_{ij}, \text{TSE2}_{ij} \)  
components of a four-pole transfer matrix for an acoustical mechanism with a sudden expanded tube

\( \text{T}_{ij}^\ast \)  
components of a four-pole transfer system matrix

\( T \)  
current temperature
\[ T_0 \quad \text{initial temperature} \]
\[ u_i \quad \text{acoustic particle velocity at the ith node (m s}^{-1}\text{)} \]
\[ u_{ij} \quad \text{acoustical particle velocity passing through a perforated hole from the ith node to the jth node (m s}^{-1}\text{)} \]
\[ V \quad \text{mean flow velocity at the inner perforated tube (m s}^{-1}\text{)} \]
\[ \rho_i \quad \text{acoustical density at the i-th node} \]
\[ \zeta \quad \text{specific acoustical impedance of the inner perforated tube} \]
\[ \sigma \quad \text{the porosity of the ith inner perforated tube.} \]
\[ \beta_i \quad \text{ith eigen value of } [N]_{i \times 4} \]
\[ [\Omega]_{4 \times 4} \quad \text{the model matrix formed by four sets of eigen vectors } \Omega_{4 \times 4} \text{ of } [N]_{4 \times 4}. \]

1. INTRODUCTION

Davis et al., started the research of mufflers in 1954 [1]. Based on the plane wave theory, studies of simple expansion mufflers without perforated holes have been discussed [2, 3]. In order to increase the acoustical performance of lower frequency sound energy, a muffler equipped with an internal perforated tube was introduced and discussed by Sullivan and Crocker in 1978 [4]. To solve the coupled equations, a series of theories and numerical techniques in decoupling the acoustical problems have been proposed [5, 6, 7]. Jayaraman and Yam [8] employed a method for finding an analytical solution by using a presumption of the unreasonable inner and outer duct in 1981. In addition, Munjal et al. [9] provided a generalized de-coupling method. Regarding the flowing effect, Peat [10] proposed the numerical decoupling method by finding the eigen value in transfer matrices. However, research work on mufflers conjugated with multiple parallel perforated tubes which may disperse the venting fluid and reduce the secondary flowing noise has been neglected.
Therefore, the main purpose of this paper is not only to analyze the sound transmission loss (STL) of two-chamber mufflers equipped with multiple parallel perforated tubes but also to optimize the best design shape within a limited space. In this paper, three kinds of two-chamber mufflers linked with multiple perforated tubes (muffler A; a two-chamber muffler equipped with one perforated tube; muffler B; a two-chamber muffler equipped with two perforated tubes; muffler C; a two-chamber muffler equipped with four perforated tubes) are introduced. Moreover, the numerical decoupling methods used in forming a four-pole system matrix are in line with the simulated annealing method.

2. THEORETICAL BACKGROUND

In this paper, three kinds of two-chamber mufflers connected with multiple perforated tubes were adopted for noise elimination in the blower room shown in Figure 1. Before the acoustical fields of the mufflers were analyzed, the acoustical elements had been recognized in advance. As shown in Figure 2, four kinds of muffler components, including four straight ducts, two sudden expanded ducts, two sudden contracted ducts, and multiple perforated ducts, are identified and marked as (I), (II), (III), and (IV). Additionally, the acoustical field within the muffler is represented by ten points. The outline dimension of the three-chamber muffler with perforated intruding tubes is shown in Figure 3. As derived in previous work [11, 12, 13, 14, 15, 16] and appendices A–C, individual transfer matrices with respect to

Complimentary Contributor Copy
straight ducts, perforated ducts, and sudden expanded/contracted ducts are described as follows:

Figure 2. Acoustical elements in three kinds of simple expanded mufflers hybridized with perforated tubes (muffler A ~ muffler C).

Figure 3. The outline dimension of three kinds of simple expanded mufflers hybridized with perforated tubes (muffler A ~ muffler C).

Complimentary Contributor Copy
2.1. Muffler A (a Two-Chamber Muffler Connected with One Perforated Tube)

As indicated in Figure 2, for the acoustical element (I), the four-pole matrix between nodes 1 and 2 is

$$\begin{bmatrix} p_1 \\ \rho_o c_o u_1 \end{bmatrix} = f_1(L_i, D_i, M_i) \begin{bmatrix} TS1_{1,1} & TS1_{1,2} \\ TS1_{2,1} & TS1_{2,2} \end{bmatrix} \begin{bmatrix} p_2 \\ \rho_o c_o u_2 \end{bmatrix}$$

(1)

For the acoustical element (II), the four-pole matrix between nodes 2 and 3 is

$$\begin{bmatrix} p_2 \\ \rho_o c_o u_2 \end{bmatrix} = \begin{bmatrix} TSE1_{1,1} & TSE1_{1,2} \\ TSE1_{2,1} & TSE1_{2,2} \end{bmatrix} \begin{bmatrix} p_3 \\ \rho_o c_o u_3 \end{bmatrix}$$

(2)

Similarly, the four-pole matrix between nodes 3 and 4 is

$$\begin{bmatrix} p_3 \\ \rho_o c_o u_3 \end{bmatrix} = f_2(L_i, D_i, M_i) \begin{bmatrix} TS2_{1,1} & TS2_{1,2} \\ TS2_{2,1} & TS2_{2,2} \end{bmatrix} \begin{bmatrix} p_4 \\ \rho_o c_o u_4 \end{bmatrix}$$

(3)

For the acoustical element (III), the four-pole matrix between nodes 4 and 5 is

$$\begin{bmatrix} p_4 \\ \rho_o c_o u_4 \end{bmatrix} = \begin{bmatrix} TSC1_{1,1} & TSC1_{1,2} \\ TSC1_{2,1} & TSC1_{2,2} \end{bmatrix} \begin{bmatrix} p_5 \\ \rho_o c_o u_5 \end{bmatrix}$$

(4)

As in the previous work [11, 12, 13, 14, 15, 16] and the derivation in Appendix A, for an acoustical element (IV) with one perforated tube, the four-pole matrix between nodes 5 and 6 is

$$\begin{bmatrix} p_5 \\ \rho_o c_o u_5 \end{bmatrix} = \begin{bmatrix} TPI_{1,1} & TPI_{1,2} \\ TPI_{2,1} & TPI_{2,2} \end{bmatrix} \begin{bmatrix} p_6 \\ \rho_o c_o u_6 \end{bmatrix}$$

(5)

Likewise, the four-pole matrix between nodes 6 and 7 for a sudden expanded duct is

Complimentary Contributor Copy
The four-pole matrix between nodes 7 and 8 in a straight duct is

\[
\begin{bmatrix}
P_6 \\
\rho_o c_o u_6
\end{bmatrix} =
\begin{bmatrix}
TSE_{2,1,1} & TSE_{2,1,2} \\
TSE_{2,2,1} & TSE_{2,2,2}
\end{bmatrix}
\begin{bmatrix}
P_7 \\
\rho_o c_o u_7
\end{bmatrix}
\]  \hspace{1cm} (6)

The four-pole matrix between nodes 8 and 9 for a sudden contracted duct is

\[
\begin{bmatrix}
P_8 \\
\rho_o c_o u_8
\end{bmatrix} =
\begin{bmatrix}
TSC_{2,1,1} & TSC_{2,1,2} \\
TSC_{2,2,1} & TSC_{2,2,2}
\end{bmatrix}
\begin{bmatrix}
P_9 \\
\rho_o c_o u_9
\end{bmatrix}
\]  \hspace{1cm} (7)

Moreover, the four-pole matrix between nodes 9 and 10 in a straight duct is

\[
\begin{bmatrix}
P_9 \\
\rho_o c_o u_9
\end{bmatrix} =
\begin{bmatrix}
TSC_{4,1,1} & TSC_{4,1,2} \\
TSC_{4,2,1} & TSC_{4,2,2}
\end{bmatrix}
\begin{bmatrix}
P_{10} \\
\rho_o c_o u_{10}
\end{bmatrix}
\]  \hspace{1cm} (8)

The total transfer matrix assembled by multiplication is

\[
\begin{bmatrix}
P_{10} \\
\rho_o c_o u_{10}
\end{bmatrix} =
\left[
\begin{array}{c|c}
T_{11}^* & T_{12}^* \\
T_{21}^* & T_{22}^*
\end{array}
\right]
\begin{bmatrix}
P_7 \\
\rho_o c_o u_7
\end{bmatrix}
\]  \hspace{1cm} (10)

A simplified form in a matrix is expressed as

\[
\begin{bmatrix}
P_1 \\
\rho_o c_o u_1
\end{bmatrix} =
\left[
\begin{array}{c|c}
T_{11}^* & T_{12}^* \\
T_{21}^* & T_{22}^*
\end{array}
\right]
\begin{bmatrix}
P_{10} \\
\rho_o c_o u_{10}
\end{bmatrix}
\]  \hspace{1cm} (11)
2.2. Muffler B (a Two-Chamber Muffler Connected with Two Perforated Tubes)

Similarly, the acoustical four-pole matrix between nodes 1~2, nodes 2~3, nodes 7~8, nodes 8~9, and nodes 9~10 is the same as Eqs.(1)~(3) and Eqs.(7)~(9). As derived in Appendix B, for two parallel perforated tubes connected to two chambers, the four-pole matrices between nodes 5\(_{(1)}\)~6\(_{(1)}\) and nodes 5\(_{(2)}\)~6\(_{(2)}\) can be combined into an equivalent matrix.

\[
\begin{bmatrix}
1 & \frac{1}{2} \cdot T P I_{1,2} \\
2 & T P I_{2,2}
\end{bmatrix}
\begin{bmatrix}
p_5 \\
p_6
\end{bmatrix}
= \begin{bmatrix}
p_5 \\
p_6
\end{bmatrix}
\]

(12)

where [TP1\(_{i,j}\)] is the four-pole matrix for a single perforated tube. Meanwhile, the equivalent acoustical field of muffler B is shown in Figure 4. Consequently, the total transfer matrix assembled by multiplication is

\[
\begin{bmatrix}
p_1 \\
p_6
\end{bmatrix} = \begin{bmatrix}
f_1(L_1, D_1, M_1) & f_2(L_1, D_1, M_1) & f_3(L_1, D_1, M_1) & f_4(L_1, D_1, M_1) \\
T S I_{1,1} & T S I_{1,2} & T S I_{1,2} & T S I_{1,2} \\
T S I_{1,1} & T S I_{1,2} & T S I_{1,2} & T S I_{1,2} \\
T S I_{2,1} & T S I_{2,2} & T S I_{2,2} & T S I_{2,2}
\end{bmatrix}
\begin{bmatrix}
1 & \frac{1}{2} \cdot T P I_{1,2} \\
2 & T P I_{2,2}
\end{bmatrix}
\begin{bmatrix}
T S E_{2,1} & T S E_{2,1} & T S E_{2,1} & T S E_{2,1} \\
T S E_{2,1} & T S E_{2,1} & T S E_{2,1} & T S E_{2,1} \\
T S E_{2,1} & T S E_{2,1} & T S E_{2,1} & T S E_{2,1} \\
T S E_{2,1} & T S E_{2,1} & T S E_{2,1} & T S E_{2,1}
\end{bmatrix}
\begin{bmatrix}
T S C_{1,1} & T S C_{1,2} & T S C_{1,2} & T S C_{1,2} \\
T S C_{1,1} & T S C_{1,2} & T S C_{1,2} & T S C_{1,2} \\
T S C_{1,1} & T S C_{1,2} & T S C_{1,2} & T S C_{1,2} \\
T S C_{1,1} & T S C_{1,2} & T S C_{1,2} & T S C_{1,2}
\end{bmatrix}
\begin{bmatrix}
p_1 \\
p_6
\end{bmatrix}
\]

(13)

A simplified form in a matrix is expressed as

\[
\begin{bmatrix}
p_1 \\
p_6
\end{bmatrix} = \begin{bmatrix}
T_{11}^{**} & T_{12}^{**} \\
T_{21}^{**} & T_{22}^{**}
\end{bmatrix}
\begin{bmatrix}
p_{10} \\
p_{10}
\end{bmatrix}
\]

(14)

Figure 4. Equivalent acoustical field for a muffler hybridized with perforated tubes (muffler B and muffler C).
2.3. Muffler C (a Two-Chamber Muffler Connected with Four Perforated Tubes)

Likewise, the acoustical four-pole matrix between nodes 1~2, nodes 2~3, nodes 7~8, nodes 8~9, and nodes 9~10 is the same as Eqs.(1)~(3) and Eqs.(7)~(9). As derived in Appendix C, for a two-chamber muffler connected with four parallel perforated tubes, the four-pole matrices between nodes 5\(^{(1)}\)~6\(^{(1)}\), 5\(^{(2)}\)~6\(^{(2)}\), 5\(^{(3)}\)~6\(^{(3)}\), and 5\(^{(4)}\)~6\(^{(4)}\) nodes can be combined into an equivalent matrix.

\[
\begin{bmatrix}
P_5 \\
\rho_o c_o u_5
\end{bmatrix} = \begin{bmatrix}
TP_{11,1} & \frac{1}{4} TP_{11,2} \\
4 \cdot TP_{22,2} & TP_{22,2}
\end{bmatrix}
\begin{bmatrix}
P_6 \\
\rho_o c_o u_6
\end{bmatrix}
\]  

(15)

where \([TP_{11}]\) is the four-pole matrix for a single perforated tube.

The related equivalent acoustical field of muffler C is also presented and shown in Figure 4. Consequently, the total transfer matrix assembled by multiplication is

\[
\begin{bmatrix}
P_1 \\
\rho_o c_o u_1
\end{bmatrix} = \begin{bmatrix}
T_{11}^{**} & T_{12}^{**} \\
T_{21}^{**} & T_{22}^{**}
\end{bmatrix}
\begin{bmatrix}
P_{10} \\
\rho_o c_o u_{10}
\end{bmatrix}
\]  

(16)

A simplified form in a matrix is expressed as

\[
\begin{bmatrix}
P_1 \\
\rho_o c_o u_1
\end{bmatrix} = \begin{bmatrix}
\rho_{0o} \rho_{c0} u_{10}
\end{bmatrix}
\]

(17)

2.4. Overall Sound Power Level

The sound transmission loss (STL) of mufflers A~C is defined as [17]
\( STL_1(Q, f, RT_1, RT_2, RT_3, RT_4, RT_5, RT_6) \)
\[ = 20\log \left( \frac{T_{11}^* + T_{12}^* + T_{23}^* + T_{22}^*}{2} \right) + 10\log \left( \frac{S_n}{S_{10}} \right) \]  
\( (18a) \)

\( STL_2(Q, f, RT_1, RT_2, RT_3, RT_4, RT_5, RT_6) \)
\[ = 20\log \left( \frac{T_{11}^* + T_{12}^* + T_{23}^* + T_{22}^*}{2} \right) + 10\log \left( \frac{S_n}{S_{10}} \right) \]  
\( (18b) \)

\( STL_3(Q, f, RT_1, RT_2, RT_3, RT_4, RT_5, RT_6) \)
\[ = 20\log \left( \frac{T_{11}^* + T_{12}^* + T_{23}^* + T_{22}^*}{2} \right) + 10\log \left( \frac{S_n}{S_{10}} \right) \]  
\( (18c) \)

where

\[ \text{RT1=}D_2; \text{RT2=}D_3/D_2; \text{RT3=}D_4; \text{RT4=}L_3; \text{RT5=}dH; \text{RT6=}\sigma \]  
\( (18d) \)

The silenced octave sound power level emitted from a muffler’s outlet is

\[ SWL_i = SWLO(f_i) - STL(f_i) \]  
\( (19) \)

where

(1) \( SWLO(f_i) \) is the original \( SWL \) at the inlet of a muffler (or pipe outlet), and \( f_i \) is the relative octave band frequency.

(2) \( STL(f_i) \) is the muffler’s \( STL \) with respect to the relative octave band frequency (\( f_i \)).

(3) \( SWL \) is the silenced \( SWL \) at the outlet of a muffler with respect to the relative octave band frequency.

Finally, the overall \( SWL_T \) silenced by a muffler at the outlet is

\[ SWL_T = 10^\log \left( \sum_{i=1}^{n} \left[ \frac{SWLO(f_i)^{10}}{STL(f_i)^{10}} + \frac{SWLO(f_i)^{10}}{STL(f_i)^{10}} + \frac{SWLO(f_i)^{10}}{STL(f_i)^{10}} + \frac{SWLO(f_i)^{10}}{STL(f_i)^{10}} + \frac{SWLO(f_i)^{10}}{STL(f_i)^{10}} \right] \right) \]  
\( (20) \)
2.5. Objective Function

(A) STL Maximization for a Tone (f) Noise

The objective functions in maximizing the STL at a pure tone (f) are

\[ OBJ_{11} = STL_1(Q, f, RT_1, RT_2, RT_3, RT_4, RT_5, RT_6) \]  
(21a)

\[ OBJ_{12} = STL_2(Q, f, RT_1, RT_2, RT_3, RT_4, RT_5, RT_6) \]  
(21b)

\[ OBJ_{13} = STL_3(Q, f, RT_1, RT_2, RT_3, RT_4, RT_5, RT_6) \]  
(21c)

The related ranges of the parameters are

\[ Q = 0.01 \text{ m}^3/\text{s}; \quad L_o = 1.2 \text{ m}; \quad D_o = 0.6 \text{ m}; \quad L_1 = 0.05 \text{ m}; \quad L_5 = 0.05 \text{ m}; \quad D_1 = 0.2 \text{ m}; \]

RT1: [0.1, 0.25]; RT2: [0.3, 0.7]; RT3: [0.1, 0.5]; RT4: [0.1, 0.5];

RT5: [0.00175, 0.007]; RT6: [0.01, 0.3]  
(21d)

(B) SWL Minimization for a Broadband Noise

To minimize the overall SWL, the objective function is

\[ OBJ_{21} = SWL_{T-1}(Q, RT_1, RT_2, RT_3, RT_4, RT_5, RT_6) \]  
(22a)

\[ OBJ_{22} = SWL_{T-2}(Q, RT_1, RT_2, RT_3, RT_4, RT_5, RT_6) \]  
(22b)

\[ OBJ_{23} = SWL_{T-3}(Q, RT_1, RT_2, RT_3, RT_4, RT_5, RT_6) \]  
(22c)

3. Model Check

Before performing the SA optimal simulation on mufflers, an accuracy check of the mathematical model on a one perforated tube is performed by Sullivan and Crocker [4]. As indicated in Figure 5, accuracy between the theoretical data and the experimental data is in agreement. Therefore, the model of two mufflers connected with multiple perforated tubes is adopted in the following optimization process.
[Experiment data is from Sullivan and Crocker [4].]

Figure 5. Performance of single-chamber perforated muffler without the mean flow \( [D_1=0.058\text{(m)}, D_0=0.0762\text{(m)}, L_c=0.0667\text{(m)}, t=0.0081\text{(m)}, dh=0.00249\text{(m)}, \eta=0.037] \).

### 4. CASE STUDIES

In this paper, the noise reduction of a blower room within a space-constrained room is exemplified and shown in Figure 1. The sound power level (SWL) inside the blower’s outlet is shown in Table 1 where the overall SWL reaches 107.4 dB.

It is obvious that five octave-band frequencies (125 Hz, 250 Hz, 500 Hz, 1000 Hz, and 2000 Hz) have higher noise levels (90~104 dB). To reduce the huge venting noise emitted from the blower’s outlet, noise elimination of the five primary noises using the two-chamber mufflers connected with multiple perforated tubes (mufflers A~C) which may disperse the venting fluid to decrease the secondary flowing noise is considered.

<table>
<thead>
<tr>
<th>f(Hz)</th>
<th>125</th>
<th>250</th>
<th>500</th>
<th>1k</th>
<th>2k</th>
<th>overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>SWL-dB(A)</td>
<td>90</td>
<td>95</td>
<td>104</td>
<td>102</td>
<td>100</td>
<td>107.4</td>
</tr>
</tbody>
</table>

Complimentary Contributor Copy
To obtain the best acoustical performance within a fixed space, numerical assessments linked to a SA optimizer are applied. Before the minimization of a broadband noise is performed, a reliability check of the SA method by maximization of the STL at a targeted one tone (300 Hz) has been performed. As shown in Figure 1, the available space for a muffler is 0.6 m in width, 0.6 m in height, and 1.2 m in length. The flow rate ($Q$) and thickness of a perforated tube ($t$) are preset at 0.01 (m$^3$/s) and 0.001(m), respectively. The corresponding OBJ functions, space constraints, and the ranges of design parameters are summarized in Eqs.(21)–(22).

### 5. SIMULATED ANNEALING METHOD

To efficiently search for the appropriate global solution in the engineering problem, Evolutionary Algorithms (EAs) have been widely developed for two decades. Simulated Annealing (SA) is one of the best stochastic search methods. Here, before the SA optimization is performed, there is no need to choose a starting design data which are required in classical gradient methods of EPFM, IPFM and FDM [18].

**Table 2. The pseudo-code implementing the simulated annealing heuristic**

| T:=To |
| X := Xo |
| F := F(X) |
| k := 0 |

while n < iter

Xn’ := neighbour(Xn)

$\Delta F = F(Xn') - F(Xn)$

if $\Delta F \leq 0$ then Xn’= Xn ; T’n=kk*Tn ; n:= n + 1

else

elseif random() < pb($\Delta F / C*Tn$) then

Xn’ = Xn ; T’n=kk*Tn ; n := n + 1

return

Complimentary Contributor Copy
Therefore, simulated annealing SA is adopted as an optimizer and used in the muffler’s shape optimization. The fundamental theory of the SA was first introduced by Metropolis et al. [19] and developed by Kirkpatrick et al. [20]. The scheme of SA is a variation of the hill-climbing algorithm.

All downhill movements for improvement are accepted for the decrement of the system’s energy. In order to escape from the local optimum, SA also allows movement resulting in solutions that are inferior (uphill moves) to the current solution. The pseudo-code implementing the simulated annealing heuristic algorithm is listed in Table 2. To emulate the SA’s evolution, a new random solution (X’) is chosen from the neighborhood of the current solution (X). If the change in the objective function (or energy) is negative (ΔF ≤ 0), the new solution will be acknowledged as the new current solution with the transition property (pb(X’)) of 1.

If the change is not negative (ΔF > 0), the probability of making the transition to the new state X’ will be the function pb(ΔF/CT). The new transition property (pb(X’)) varied from 0–1 will be calculated by the Boltzmann’s factor (pb(X’) = exp(ΔF / CT)) as shown in Eq. (23)

\[
pb(X') = \begin{cases} 
1, & \Delta F \leq 0 \\
\exp\left(-\frac{\Delta F}{CT}\right), & \Delta f > 0 
\end{cases} \tag{23a}
\]

\[
\Delta F = F(X') - F(X) \tag{23b}
\]

where C and T are the Boltzmann constant and the current temperature. Here, if the transition property [pb(X’)] is greater than a random number of rand (0,1), the new solution (worse solution) which results in a higher energy condition will then be accepted; otherwise, it will be rejected. Each successful substitution of the new current solution will conduct to the decay of the current temperature as

\[
T_{\text{new}} = kk \times T_{\text{old}} \tag{24}
\]

where kk is the cooling rate.

The process is repeated until the preset (iter) of the outer loop is reached.

Complimentary Contributor Copy
6. RESULTS AND DISCUSSION

6.1. Results

The accuracy of the SA optimization depends on two kinds of SA parameters including \( kk \) (cooling rate) and \( \text{iter} \) (maximum iteration). To achieve good optimization, the following parameters are varied step by step

\[ \text{iter} (50, 100, 500, 1000, 2000); \text{kk} (0.91, 0.93, 0.95, 0.97, 0.99). \]

Two results of optimization (one, a pure tone noise used for SA’s accuracy check; and the other, a broadband noise occurring in a blower room) are described below.

6.1.1. Pure Tone Noise Optimization

Before dealing with a broadband noise, the STL’s maximization with respect to a one-tone noise (300Hz) is introduced for a reliability check on the SA method. By using Eqs. (18a)(21b), the maximization of the STL with respect to muffler A (a two-chamber muffler hybridized with one perforated tube) at the specified pure tone (300Hz) was performed first. As indicated in Table 3, nine sets of SA parameters are tried in the muffler’s optimization. Obviously, the optimal design data can be obtained from the last set of SA parameters at \((\text{kk, iter}) = (0.99, 2000)\). Using the optimal design in a theoretical calculation, the optimal STL curves with respect to various SA parameters \((\text{kk, iter})\) are plotted and depicted in Figures 6 and 7.

Table 3. Optimal STL for muffler A (equipped with one perforated tube) at various SA parameters (targeted tone of 300 Hz)

<table>
<thead>
<tr>
<th>SA parameter</th>
<th>Design parameters</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>iter</td>
<td>kk</td>
<td>RT1</td>
</tr>
<tr>
<td>50</td>
<td>0.91</td>
<td>0.2328</td>
</tr>
<tr>
<td>50</td>
<td>0.93</td>
<td>0.2075</td>
</tr>
<tr>
<td>50</td>
<td>0.95</td>
<td>0.1889</td>
</tr>
<tr>
<td>50</td>
<td>0.97</td>
<td>0.1537</td>
</tr>
<tr>
<td>50</td>
<td>0.99</td>
<td>0.1399</td>
</tr>
<tr>
<td>100</td>
<td>0.99</td>
<td>0.1171</td>
</tr>
<tr>
<td>500</td>
<td>0.99</td>
<td>0.1107</td>
</tr>
<tr>
<td>1000</td>
<td>0.99</td>
<td>0.1066</td>
</tr>
<tr>
<td>2000</td>
<td>0.99</td>
<td>0.1042</td>
</tr>
</tbody>
</table>

Complimentary Contributor Copy
As revealed in Figures 6 and 7, the STL is precisely maximized at the desired frequency. Consequently, the SA optimizer is reliable in the optimization process.
6.1.2. Broadband Noise Optimization

Similarly, considering Eqs. (21a-d) and optimizing using the SA method, the minimization of the $SWL_{T,1}$, $SWL_{T,2}$, and $SWL_{T,3}$ with respect to mufflers A–C was performed by adjusting the SA parameters. As indicated in Table 4, eight sets of SA parameters are tried in the muffler’s optimization. Obviously, the optimal design data can be obtained from the last set of SA parameters at $(kk, \text{iter}) = (0.99, 1000)$.

Table 4. Optimal STL for muffler A (equipped with one perforated tube) at various SA parameters (broadband noise)

<table>
<thead>
<tr>
<th>SA parameter</th>
<th>Design parameters</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>iter, kk</td>
<td>RT1, RT2, RT3, RT4, RT5, RT6</td>
<td>$SWL_T$ – dB(A)</td>
</tr>
<tr>
<td>50, 0.91</td>
<td>0.1772, 0.5058, 0.3058, 0.4029, 0.004451, 0.1592</td>
<td>101.2</td>
</tr>
<tr>
<td>50, 0.93</td>
<td>0.2248, 0.6329, 0.4329, 0.4664, 0.006199, 0.2513</td>
<td>99.2</td>
</tr>
<tr>
<td>50, 0.95</td>
<td>0.1587, 0.4566, 0.2566, 0.3783, 0.003806, 0.1236</td>
<td>93.7</td>
</tr>
<tr>
<td>50, 0.97</td>
<td>0.1406, 0.4082, 0.2082, 0.3541, 0.003171, 0.08847</td>
<td>82.3</td>
</tr>
<tr>
<td>50, 0.99</td>
<td>0.1239, 0.3636, 0.1636, 0.3318, 0.002585, 0.05612</td>
<td>69.6</td>
</tr>
<tr>
<td>100, 0.99</td>
<td>0.1188, 0.3501, 0.1501, 0.3251, 0.002408, 0.04636</td>
<td>65.3</td>
</tr>
<tr>
<td>500, 0.99</td>
<td>0.1102, 0.3271, 0.1271, 0.3135, 0.002106, 0.02964</td>
<td>56.6</td>
</tr>
<tr>
<td>1000, 0.99</td>
<td>0.1047, 0.3125, 0.1125, 0.3063, 0.001914, 0.01906</td>
<td>49.7</td>
</tr>
</tbody>
</table>

Figure 8. $STL$ with respect to various $kk$ [muffler A: iter=50, broadband noise].
Using the optimal design data in a theoretical calculation, the optimal STL curves with respect to various SA parameters (kk, iter) are plotted and depicted in Figures 8 and 9.

By using the above SA parameters, the muffler’s optimal design data for muffler B and muffler C used to minimize the sound power level at the muffler’s outlet is summarized in Table 5. As illustrated in Table 5, the resultant sound power levels with respect to three kinds of mufflers have been reduced from 107.4 dB(A) to 49.7 dB(A), 54.5 dB(A), and 59.5 dB(A). Using this optimal design in a theoretical calculation, the optimal STL curves with respect to various mufflers are plotted and compared with the original SWL depicted in Figure 10.

**Table 5. Comparison of the minimized SWL_T of three kinds of mufflers (mufflers A–C) [broadband noise]**

<table>
<thead>
<tr>
<th>Muffler Type</th>
<th>Design parameters</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RT1</td>
<td>RT2</td>
</tr>
<tr>
<td>Muffler A</td>
<td>0.1047</td>
<td>0.3125</td>
</tr>
<tr>
<td>Muffler B</td>
<td>0.1047</td>
<td>0.3125</td>
</tr>
<tr>
<td>Muffler C</td>
<td>0.1047</td>
<td>0.3125</td>
</tr>
</tbody>
</table>
6.2. Discussion

In order to decrease a secondary flowing noise generated from a higher speed flow, new muffler designs with multiple perforated tubes are presented. To achieve a sufficient optimization, the selection of the appropriate \( S_A \) parameter set is essential. As indicated in Table 3, the best \( S_A \) set of muffler A at the targeted pure tone noise of 300 Hz has been shown. The related \( STL \) curves with respect to various \( S_A \) parameters are plotted in Figures 6 and 7. Figures 6 and 7 reveal the predicted maximal value of the \( STL \) is located at the desired frequency. In dealing with the broadband noise, the acoustical performance among three kinds of two-chamber mufflers connected with multiple perforated tubes (mufflers A, B, and C) are shown in Table 5, Figure 8, Figure 9, and Figure 10. As can be observed in Table 5, the overall sound transmission loss of the two-chamber muffler equipped with one perforated tube (muffler A) reaches 57.8 dB. However, the overall sound transmission losses of the two-chamber mufflers equipped with two perforated tubes (muffler B) and four perforated tubes (muffler C) are 52.9 dB and 47.9 dB.
The results shown in Table 5 and Figure 10 indicate that the two-chamber muffler hybridized with the least number of perforated tubes is superior to the other mufflers equipped with more perforated tubes. It can be seen that for the mufflers equipped with more perforated tubes, the acoustical performance of the acoustical element (II) between nodes 4 and 5 and acoustical element (III) between nodes 6 and 7 will largely decrease due to the decrement of the area ratio; therefore, the overall noise reduction of the mufflers with more perforated tubes will decrease.

**CONCLUSION**

It has been shown that two-chamber mufflers hybridized with multiple perforated tubes can be easily and efficiently optimized within a limited space by using a decoupling technique, a plane wave theory, a four-pole transfer matrix, and a SA optimizer. As indicated in Table 3, Table 4, and Figures 6~9, two kinds of SA parameters ($kk$ and $iter$) play essential roles in the solution’s accuracy during SA optimization. Figures 6~9 indicate that the tuning ability established by adjusting design parameters of muffler A is reliable.

In addition, the appropriate acoustical performance curve of three kinds of two-chamber mufflers hybridized with multiple perforated tubes (mufflers A~C) has been assessed. As indicated in Table 5, the resultant $SWL_T$ with respect to these mufflers is 49.7 dB(A), 54.5 dB(A), and 59.5 dB(A). Obviously, the two-chamber muffler hybridized with the least number of perforated tubes is superior to the other mufflers equipped with more perforated tubes. It can be seen that more perforated tubes installed between the two-chamber muffler will disperse the venting fluid and reduce the secondary flowing noise; however, the acoustical performance of the acoustical element (II) between nodes 4 and 5 and acoustical element (III) between nodes 6 and 7 will largely decrease even though the number of the parallel perforated tubes for the muffler increases. Therefore, before choosing the appropriate number of perforated tubes and performing the muffler’s shape optimization, presetting the maximum flow velocity within the muffler is required.
APPENDIX A

Transfer Matrix of an Perforated Tube

As indicated in Figure 11, the perforated resonator is composed of an inner perforated tube and an outer resonating chamber.

\[
\begin{align*}
\text{Inner tube} & \\
\text{continuity equation} & \frac{\partial \rho_S}{\partial x} + \rho_o \frac{\partial u_S}{\partial x} + 4 \frac{\rho_o}{D_o} u + \frac{\partial p_S}{\partial t} = 0 \tag{A1} \\
\text{momentum equation} & \rho_o \left( \frac{\partial}{\partial t} + V \frac{\partial}{\partial x} \right) u_S + \frac{\partial p_S}{\partial x} = 0 \tag{A2}
\end{align*}
\]

\[\text{Outer tube}\]
continuity equation

Complimentary Contributor Copy
\[ \rho_o \frac{\partial u_{sa}}{\partial x} - \frac{4D_2 \rho_o}{D_o^2 - D_1^2} u + \frac{\partial p_{sa}}{\partial t} = 0 \]  
(A3)

momentum equation

\[ \rho_o \frac{\partial u_{sa}}{\partial t} + \frac{\partial p_{sa}}{\partial x} = 0 \]  
(A4)

Assuming the sound wave to be propagated in a harmonic motion and in an isentropic process, it has

\[ p(x,t) = P(x) \cdot e^{i\omega t} \]  
(A5a)

where

\[ P(x) = \rho(x) \cdot c_o^2 \]  
(A5b)

Plugging Eq.(A5) into the above Eqs. and rearranging them yield

\[ \rho_o c_o \frac{du_5}{dx} = \left[ jkp_5 + \frac{V}{c_o} \cdot \frac{dp_5}{dx} + \frac{4(p_5 - p_{sa})}{D_5} \right] \]  
(A6a)

\[ \rho_o c_o \frac{du_{sa}}{dx} = \left[ jkp_{sa} - \frac{4D_2}{(D_o^2 - D_1^2)} (p_5 - p_{sa}) \right] \]  
(A6b)

\[ \rho_o c_o \left( jk u_5 + \frac{V}{c_o} \cdot \frac{du_5}{dx} \right) = -\frac{dp_5}{dx} \]  
(A6c)

\[ j\rho_o c_o ku_{sa} = -\frac{dp_{sa}}{dx} \]  
(A6d)

Rearranging Eqs.(A6) by eliminating \( u_5 \) and \( u_{sa} \) yields

Complimentary Contributor Copy
\[
\left(1 - M^2 S\right) \frac{d^2}{dx^2} - 2 jM k \frac{d}{dx} + k^2 \right] p_5 - \frac{4}{D_0^2} \left[ M \frac{d}{dx} + jk \right] (p_5 - p_{5a}) = 0 \quad (A7)
\]

\[
\frac{d^2}{dx^2} + k^2 \right] p_{5a} + j \frac{4D_2}{(D_0^2 - D_2^2)} (p_5 - p_{5a}) = 0 \quad (A8)
\]

Eqs. (A7) and (A8) can be expressed in a matrix form as

\[
\begin{bmatrix}
D^2 + \alpha_1 D + \alpha_2 & \alpha_3 D + \alpha_4 \\
\alpha_3 D + \alpha_6 & D^2 + \alpha_5 D + \alpha_8
\end{bmatrix}
\begin{bmatrix}
p_5 \\
p_{5a}
\end{bmatrix}
= \begin{bmatrix}
0 \\
0
\end{bmatrix} \quad (A9)
\]

Developing Eq. (A9) yields

\[
p_5^{\prime\prime} + \alpha_4 p_5^{\prime} + \alpha_2 p_5 + \alpha_3 p_{5a}^{\prime} + \alpha_4 p_{5a} = 0 \quad (A10a)
\]

\[
\alpha_5 p_5^{\prime} + \alpha_6 p_5 + p_{5a}^{\prime\prime} + \alpha_7 p_{5a}^{\prime} + \alpha_8 p_{5a} = 0 \quad (A10b)
\]

Setting \( p_5^{\prime} = \frac{dp_5}{dx} = y_1, \ p_{5a}^{\prime} = \frac{dp_{5a}}{dx} = y_2, \ p_5 = y_3, \) and \( p_{5a} = y_4, \) the new matrix form yields

\[
\begin{bmatrix}
y_1 \\
y_2 \\
y_3 \\
y_4
\end{bmatrix}
= \begin{bmatrix}
-\alpha_1 & -\alpha_3 & -\alpha_2 & -\alpha_4 \\
-\alpha_3 & -\alpha_7 & -\alpha_6 & -\alpha_8 \\
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
y_1 \\
y_2 \\
y_3 \\
y_4
\end{bmatrix} \quad (A11)
\]

which can be briefly expressed as

\[
\{y^{\prime}\} = [N]\{y\} \quad (A12)
\]

Let \( \{y\} = [O]\{\Gamma\} \quad (A13a) \]

which is

Complimentary Contributor Copy
\[
\begin{bmatrix}
\frac{dp_1}{dx} \\
\frac{dp_2}{dx} \\
p_5 \\
p_5
\end{bmatrix} =
\begin{bmatrix}
\Omega_{1,1} & \Omega_{1,2} & \Omega_{1,3} & \Omega_{1,4} \\
\Omega_{2,1} & \Omega_{2,2} & \Omega_{2,3} & \Omega_{2,4} \\
\Omega_{3,1} & \Omega_{3,2} & \Omega_{3,3} & \Omega_{3,4} \\
\Omega_{4,1} & \Omega_{4,2} & \Omega_{4,3} & \Omega_{4,4}
\end{bmatrix}
\begin{bmatrix}
\Gamma_1 \\
\Gamma_2 \\
\Gamma_3 \\
\Gamma_4
\end{bmatrix}
\] (A13b)

\[\begin{bmatrix}
\Omega
\end{bmatrix}_{4 \times 4}
\text{is the model matrix formed by four sets of eigen vectors } \Omega_{4,i} \text{ of } \begin{bmatrix} N \end{bmatrix}_{4 \times 4}.
\]

Substituting Eq.(A13) into (A12) and then multiplying \([\Omega]^{-1}\) by both sides yield

\[\begin{bmatrix}
\Omega
\end{bmatrix}^{-1}\begin{bmatrix}
\Omega
\end{bmatrix}\Gamma = \begin{bmatrix}
\Omega
\end{bmatrix}^{-1}\begin{bmatrix} N \end{bmatrix}\begin{bmatrix}
\Omega
\end{bmatrix}\Gamma
\] (A14)

Set

\[\begin{bmatrix}
x
\end{bmatrix} = \begin{bmatrix}
\Omega
\end{bmatrix}^{-1}\begin{bmatrix} N \end{bmatrix}\begin{bmatrix}
\Omega
\end{bmatrix} =
\begin{bmatrix}
\beta_1 & 0 & 0 & 0 \\
0 & \beta_2 & 0 & 0 \\
0 & 0 & \beta_3 & 0 \\
0 & 0 & 0 & \beta_4
\end{bmatrix}
\] (A15)

where \(\beta_i\) is the eigen value of \([N]\).

Eq.(A13) can be thus rewritten as

\[\begin{bmatrix}
\Gamma
\end{bmatrix} = \begin{bmatrix}
x
\end{bmatrix}\Gamma
\] (A16)

Obviously, Eq.(A16) is a decoupled equation. The related solution then becomes

\[\Gamma_i = f_i e^\beta_i x
\] (A17)

Using Eqs.(A2),(A4),(A13) and (A17), the relationship of acoustic pressure and particle velocity becomes
where

\[
H_{i,j} = \Omega_{2,i} e^{B_j x}; H_{2,i} = \Omega_{4,i} e^{B_j x}; H_{3,j} = -\frac{\Omega_{4,i} e^{B_j x}}{jk + \beta_i}; H_{4,j} = -\frac{\Omega_{2,i} e^{B_j x}}{jk}
\]  

(A18b)

Substituting \(x=0\) and \(x=Lc\) into Eq.(A18) yields

\[
\begin{bmatrix}
 p_s(0) \\
p_{sa}(0) \\
\rho_c c_u_s(0) \\
\rho_c c_u_{sa}(0)
\end{bmatrix} = [H(0)] \begin{bmatrix}
f_1 \\
f_2 \\
f_3 \\
f_4
\end{bmatrix};
\]

(A19a)

\[
\begin{bmatrix}
p_s(Lc) \\
p_{sa}(Lc) \\
\rho_c c_u_s(Lc) \\
\rho_c c_u_{sa}(Lc)
\end{bmatrix} = [H(Lc)] \begin{bmatrix}
f_1 \\
f_2 \\
f_3 \\
f_4
\end{bmatrix}
\]

(A19b)

Combining with Eqs.(A19a) and (A19b), the resultant relationship of acoustic pressure and particle velocity between \(x=0\) and \(x=Lc\) becomes

\[
\begin{bmatrix}
p_s(0) \\
p_{sa}(0) \\
\rho_c c_u_s(0) \\
\rho_c c_u_{sa}(0)
\end{bmatrix} = [\mathbb{T}][H(Lc)]^{-1} \begin{bmatrix}
p_s(Lc) \\
p_{sa}(Lc) \\
\rho_c c_u_s(Lc) \\
\rho_c c_u_{sa}(Lc)
\end{bmatrix}
\]

(A20a)

where

\[
[\mathbb{T}] = [H(0)]^{-1} [H(Lc)]^{-1}
\]

(A20b)

Complimentary Contributor Copy
Rearranging above Eqs. yields
\[
\begin{bmatrix}
  p_5(0) \\
  \rho_o c_o u_5(0)
\end{bmatrix} =
\begin{bmatrix}
  TP_{1,1} & TP_{1,2} \\
  TP_{2,1} & TP_{2,2}
\end{bmatrix}
\begin{bmatrix}
  p_5(L_c) \\
  \rho_o c_o u_5(L_c)
\end{bmatrix} \tag{A21}
\]

Let
\[
p_5(0) = p_5; p_5(L_c) = p_6; u_5(0) = u_5; u_5(L_c) = u_6 \tag{A22}
\]

It becomes
\[
\begin{bmatrix}
  p_5 \\
  \rho_o c_o u_5
\end{bmatrix} =
\begin{bmatrix}
  TP_{1,1} & TP_{1,2} \\
  TP_{2,1} & TP_{2,2}
\end{bmatrix}
\begin{bmatrix}
  p_6 \\
  \rho_o c_o u_6
\end{bmatrix} \tag{A23}
\]

**APPENDIX B**

**Transfer Matrix of Two Parallel Perforated Tubes**

As indicated in Figure 3, muffler B’s acoustical four-pole matrix between node 5(1) and 6(1) is
\[
\begin{bmatrix}
  P_{5(1)} \\
  \rho_o c_o u_{5(1)}
\end{bmatrix} =
\begin{bmatrix}
  TP_{5,1,1} & TP_{5,1,2} \\
  TP_{5,2,1} & TP_{5,2,2}
\end{bmatrix}
\begin{bmatrix}
  p_{6(1)} \\
  \rho_o c_o u_{6(1)}
\end{bmatrix} \tag{B1}
\]

Developing Eq.(B1) yields
\[
p_{5(1)} = TP_{5,1,1} \cdot p_{6(1)} + TP_{5,1,2} \cdot \rho_o c_o u_{6(1)} \tag{B2a}
\]
\[
\rho_o c_o u_{5(1)} = TP_{5,2,1} \cdot p_{6(1)} + TP_{5,2,2} \cdot \rho_o c_o u_{6(1)} \tag{B2b}
\]

Similarly, the acoustical four-pole matrix between node 5(2) and 6(2) is
\[
\begin{bmatrix}
  P_{5(2)} \\
  \rho_o c_o u_{5(2)}
\end{bmatrix} =
\begin{bmatrix}
  TP_{5,1,1} & TP_{5,1,2} \\
  TP_{5,2,1} & TP_{5,2,2}
\end{bmatrix}
\begin{bmatrix}
  p_{6(2)} \\
  \rho_o c_o u_{6(2)}
\end{bmatrix} \tag{B3}
\]
Developing Eq.(B3) yields

\[ p_{5(2)} = TP5_{11} \cdot p_{6(2)} + TP5_{12} \cdot \rho_o c_o u_{6(2)} \]  

(B4a)

\[ \rho_o c_o u_{5(2)} = TP5_{21} \cdot p_{6(2)} + TP5_{22} \cdot \rho_o c_o u_{6(2)} \]  

(B4b)

Combining Eq.(B2a) and Eq.(B4a) yields

\[ p_{5(1)} + p_{5(2)} = TP5_{11} \cdot [p_{6(1)} + p_{6(2)}] + TP5_{12} \cdot \rho_o c_o [u_{6(1)} + u_{6(2)}] \]  

(B5)

Likewise, combining Eq.(B2b) and Eq.(B4b) yields

\[ \rho_o c_o [u_{5(1)} + u_{5(2)}] = TP5_{21} \cdot [p_{6(1)} + p_{6(2)}] + TP5_{22} \cdot \rho_o c_o [u_{6(1)} + u_{6(2)}] \]  

(B6)

where

\[ p_5 = p_{5(1)} \ ; \ p_6 = p_{6(2)} \ ; \ u_5 = u_{5(1)} + u_{5(2)} \ ; \ u_6 = u_{6(1)} + u_{6(2)} \]  

(B7)

Plugging Eq.(B7) into Eqs.(B5) and (B6) yields

\[ 2 \cdot p_5 = 2 \cdot TP5_{11} \cdot p_6 + TP5_{12} \cdot \rho_o c_o u_6 \]  

(B8a)

\[ \rho_o c_o u_5 = 2 \cdot TP5_{21} \cdot p_6 + TP5_{22} \cdot \rho_o c_o u_6 \]  

(B8b)

Rearranging Eq.(B8) in a matrix form, the equivalent four-pole matrix between nodes 5 and 6 shown in Figure 4 is

\[
\begin{bmatrix}
  p_5 \\
  \rho_o c_o u_5
\end{bmatrix} =
\begin{bmatrix}
  TP5_{11} & \frac{1}{2} \cdot TP5_{12} \\
  2 \cdot TP5_{21} & TP5_{22}
\end{bmatrix}
\begin{bmatrix}
  p_6 \\
  \rho_o c_o u_6
\end{bmatrix}
\]  

(B9)
Transfer Matrix of Four Parallel Perforated Tubes

As indicated in Figure 3, muffler C’s acoustical four-pole matrix between node 5(1) and 6(1) is

\[
\begin{bmatrix}
    p_{5(1)} \\
    \rho_c u_{5(1)}
\end{bmatrix} =
\begin{bmatrix}
    TP_{5,11} & TP_{5,12} \\
    TP_{5,21} & TP_{5,22}
\end{bmatrix}
\begin{bmatrix}
    p_{6(1)} \\
    \rho_c u_{6(1)}
\end{bmatrix}
\]  \hspace{1cm} (C1)

Developing Eq.(C1) yields

\[ p_{5(1)} = TP_{5,11} \cdot p_{6(1)} + TP_{5,12} \cdot \rho_c u_{6(1)} \]  \hspace{1cm} (C2a)

\[ \rho_c u_{5(1)} = TP_{5,21} \cdot p_{6(1)} + TP_{5,22} \cdot \rho_c u_{6(1)} \]  \hspace{1cm} (C2b)

Similarly, the acoustical four-pole matrices between node 5(2) and 6(2), node 5(3) and 6(3), and node 5(4) and 6(4), is

\[
\begin{bmatrix}
    p_{5(2)} \\
    \rho_c u_{5(2)}
\end{bmatrix} =
\begin{bmatrix}
    TP_{5,11} & TP_{5,12} \\
    TP_{5,21} & TP_{5,22}
\end{bmatrix}
\begin{bmatrix}
    p_{6(2)} \\
    \rho_c u_{6(2)}
\end{bmatrix}
\]  \hspace{1cm} (C3a)

\[
\begin{bmatrix}
    p_{5(3)} \\
    \rho_c u_{5(3)}
\end{bmatrix} =
\begin{bmatrix}
    TP_{5,11} & TP_{5,12} \\
    TP_{5,21} & TP_{5,22}
\end{bmatrix}
\begin{bmatrix}
    p_{6(3)} \\
    \rho_c u_{6(3)}
\end{bmatrix}
\]  \hspace{1cm} (C3b)

\[
\begin{bmatrix}
    p_{5(4)} \\
    \rho_c u_{5(4)}
\end{bmatrix} =
\begin{bmatrix}
    TP_{5,11} & TP_{5,12} \\
    TP_{5,21} & TP_{5,22}
\end{bmatrix}
\begin{bmatrix}
    p_{6(4)} \\
    \rho_c u_{6(4)}
\end{bmatrix}
\]  \hspace{1cm} (C3c)

Developing Eqs.(C3a)–(C3c) yields

\[ p_{5(2)} = TP_{5,11} \cdot p_{6(2)} + TP_{5,12} \cdot \rho_c u_{6(2)} \]  \hspace{1cm} (C4a)

\[ \rho_c u_{5(2)} = TP_{5,21} \cdot p_{6(2)} + TP_{5,22} \cdot \rho_c u_{6(2)} \]  \hspace{1cm} (C4b)

\[ p_{5(3)} = TP_{5,11} \cdot p_{6(3)} + TP_{5,12} \cdot \rho_c u_{6(3)} \]  \hspace{1cm} (C4c)
\[ \rho_c c u_{5(3)} = TP5_{2,1} \cdot p_{6(3)} + TP5_{2,2} \cdot \rho_c c u_{6(3)} \]  
\text{(C4d)}

\[ p_{5(4)} = TP5_{1,1} \cdot p_{6(4)} + TP5_{1,2} \cdot \rho_c c u_{6(4)} \]  
\text{(C4e)}

\[ \rho_c c u_{5(4)} = TP5_{2,1} \cdot p_{6(4)} + TP5_{2,2} \cdot \rho_c c u_{6(4)} \]  
\text{(C4f)}

Combining Eq.(C2a) and Eq.(C4a),(C4c),(C4e) yields

\[ p_{5(1)} + p_{5(2)} + p_{5(3)} + p_{5(4)} = TP5_{1,1} \cdot [p_{6(1)} + p_{6(2)} + p_{6(3)} + p_{6(4)}] + TP5_{1,2} \cdot \rho_c c [u_{6(1)} + u_{6(2)} + u_{6(3)} + u_{6(4)}] \]  
\text{(C5)}

Likewise, combining Eq.(C2b) and Eq.(C4b),(C4d), and (C4f) yields

\[ \rho_c c [u_{5(1)} + u_{5(2)} + u_{5(3)} + u_{5(4)}] = TP5_{2,1} \cdot [p_{6(1)} + p_{6(2)} + p_{6(3)} + p_{6(4)}] + TP5_{2,2} \cdot \rho_c c [u_{6(1)} + u_{6(2)} + u_{6(3)} + u_{6(4)}] \]  
\text{(C6)}

where

\[ p_5 = p_{5(1)} = p_{5(2)} = p_{5(3)} = p_{5(4)}; p_6 = p_{6(1)} = p_{6(2)} = p_{6(3)} = p_{6(4)}; \]
\[ u_5 = u_{5(1)} + u_{5(2)} + u_{5(3)} + u_{5(4)}; u_6 = u_{6(1)} + u_{6(2)} + u_{6(3)} + u_{6(4)} \]  
\text{(C7)}

Plugging Eq.(C7) into Eqs.(C5) and (C6) yields

\[ 4 \cdot p_5 = 4 \cdot TP5_{1,1} \cdot p_6 + TP5_{1,2} \cdot \rho_c c u_6 \]  
\text{(C8a)}

\[ \rho_c c u_5 = 4 \cdot TP5_{2,1} \cdot p_6 + TP5_{2,2} \cdot \rho_c c u_6 \]  
\text{(C8b)}

Rearranging Eq.(C8) in a matrix form, the equivalent four-pole matrix between nodes 5 and 6 shown in Figure 4 is

\[
\begin{bmatrix}
    p_5 \\
    \rho_c c u_5
\end{bmatrix}
= \begin{bmatrix}
    TP5_{1,1} & 1 \cdot TP5_{1,2} \\
    4 \cdot TP5_{2,1} & TP5_{2,2}
\end{bmatrix}
\begin{bmatrix}
    p_6 \\
    \rho_c c u_6
\end{bmatrix}
\]  
\text{(C9)}
ACKNOWLEDGMENTS

The author acknowledges the financial support of the National Science Council (NSC 102-2622-E-235-001-CC3, ROC)

REFERENCES


A SIMULATED ANNEALING ALGORITHM TO SOLVE THE LOG-TRUCK SCHEDULING PROBLEM

Mauricio Acuna¹* and John Sessions²

¹AFORA, University of the Sunshine Coast, Hobart, TAS, Australia
²Department of Forest Engineering, Resources and Management, Oregon State University, Corvallis, OR, US

ABSTRACT

In this paper, we present a simulated annealing (SA) approach for solving the log-truck scheduling problem, which is an extension of the timber transport vehicle routing problem with time windows (TTVRPTW). The problem is characterized by a heterogeneous fleet of trucks that depart from their corresponding depots and must perform a number of daily transport tasks to move log products between wood pickup (harvest areas) locations and industrial customers, such as pulpmills and sawmills. Commencing at their depots, each truck must perform a number of successive transport tasks, each one characterized by a trip between a designated harvest area and an industrial customer, until the number of daily working hours is completed. The objective of the model is to minimize total empty travel time for the whole fleet as well as waiting times at wood pickup locations. In addition, time

* Corresponding author: Tel.: +61-03-62375622; fax: +61-03-62267942 E-mail address: macuna@usc.edu.au.

Complimentary Contributor Copy
windows and accessibility to harvest areas and customers, must be taken into account. Our solution approach consists of three main steps: determination of a cooling scheme, search for a random feasible initial solution, and application of the simulation annealing procedure with a random neighborhood that features various types of moves, including single and multiple insertions and swaps between trucks. The efficiency of the heuristics was evaluated and validated with real-life data for two problem instances, each consisting of 30 transport tasks and a fleet of 10 trucks. Considering all the scenarios evaluated, our best SA solutions resulted in maximum deviations of 3% in comparison with the optimal solutions obtained with commercial optimization solvers.

Keywords: log-truck scheduling, simulated annealing, metaheuristics, wood transport vehicle routing and scheduling problem

1. INTRODUCTION

Worldwide, transportation from forestry harvest areas to mills costs the forestry industry millions of dollars annually, accounting for up to half of the operational costs in forestry supply chains (McDonald 2001, Acuna 2011, Audy et al., 2012). Given the level of spending on transportation costs, even small increases in efficiency can reduce costs substantially (Rönqvist 1998, Palmgren 2001, Palmgren et al., 2004). It is therefore important to organize the scheduling of trucks efficiently as they have a high initial investment cost in their purchase, and high operational cost over their daily operations and ownership lifetime.

Managing log transport is generally the responsibility of the supplier, namely, the forest owner. Transport management is performed at a number of levels, from the strategic planning of roads, to the tactical implementation of haulage and harvesting operations. In Australia and most of the United States, the operational implementation is carried out on a “stump-to-mill” or “cut-and-haul” basis, which means that individual logging contractors control the operations of logging and the delivery to the mill. This brings some management efficiencies to the logging process, but it results in inefficiencies to the transport process, especially when several types of log products must be delivered to different markets (Robinson 1995). In response to this, a more efficient, centralized truck scheduling and dispatching transportation approach has been proposed to reduce costs and improve the tactical coordination of
A Simulated Annealing Algorithm to Solve the Log-Truck …


Decisions on timber transport planning must be made at different levels and planning horizons: strategical, tactical, operational, and real time (Palmgren 2001, Rönnqvist 2003). In this paper we focus on the operational aspects of timber transportation, specifically, on the optimal determination of a set of routes and schedules that a fleet of trucks performs to deliver timber (log products) from wood pickup locations (harvest areas) to customer sites (mills). Scheduling means to plan the entire fleet’s route in advance (typically one day ahead), listing every pickup and delivery of timber products. To come up with a good scheduling plan, an estimation of the stocks of logs at the pickup locations and demand at the customer sites is required. For the implementation of the scheduling plan, each truck driver receives a copy of the schedule the night before, including arrival times at each destination. In practice, however, it is quite difficult to run a log truck fleet according to a pre-arranged schedule. Unexpected delays, breakdowns, and queues mean some logs are delivered late or not at all, all of which disturb the planned schedule. In such cases, it is essential to have dispatcher on hand to repair the schedule in cooperation with truck drivers and transport coordinators (Robinson 1996, Palmgren 2001).

The scheduling planning problem described above is referred in the literature as to the timber transport vehicle routing problem (TTRVP) (Karanta 2000, Gronalt and Hirsch 2007). The TTRVP is related to the vehicle routing problem with time windows (VRPTW), described by Laporte (1992) and Toth and Vigo (2002), and to the pickup and delivery problem with time windows (PDPTW), described by Berbeglia et al. (2007) and Ropke and Pisinger (2006). In addition, multiple depots can be added to problem, and the new formulation in known as the multiple depot vehicle routing problem with pickup and delivery (MDVRPPDTW) (Oberscheider et al., 2013). Palmgren (2001) points out that the major differences that set the log truck scheduling problem apart from the VRPTW and PDPTW, are the number of coupling constraints combinations and the possibility of visiting the same location several times during the same day. This also means that a truck can drive between several pickup and delivery points during its entire daily route. Palmgren also points out that in the truck scheduling problem a customer can be visited several times during the day until its demand is satisfied. Other specific features of the truck scheduling problem include the specific type of products being demanded by the mills, and the possibility that not all trucks in an operation can serve all pickup or delivery nodes.
Several solution approaches have been proposed to solve the truck scheduling problem (for a comprehensive review see Audy et al., 2012). Shen and Sessions (1989) proposed a network-based method to generate a daily truck schedule that meets a mill delivery program with multiple time windows. Linnainmaa et al. (1995) proposed a three-phase method that includes a heuristic to allocate supply volume to demand points and solve the weekly truck schedule, and semi-manual post processing is performed by a transportation planner for the daily routes. Weintraub et al. (1996) proposed a simulation-based method with embedded heuristic rules that assigns, on a rolling time horizon, one load at a time to available trucks and thus, generates a daily truck schedule. Palmgren et al. (2004) and Rey et al. (2009) propose a column generation method in which each column corresponds to one feasible route for a truck. These models are based on generalized set partitioning models or general column-based Mixed Integer Programming (MIP) models. In addition, several authors have used metaheuristics to solve the truck scheduling problem. Gronalt and Hirsch (2007) and El Hachemi et al. (2009) propose different strategies of the Tabu Search (TS) heuristic, while Rummukainen et al. (2009) propose a method that includes a TS heuristic to create routes and a full truckload-size request, in combination with a MIP model to allocate truckloads to demand sites. In addition, McDonald et al. (2010) proposed a simulated annealing (SA) method to generate a daily truck schedule to deliver a set of requests.

Previous work has considered the problem of optimized scheduling of log trucks and thus a number of truck scheduling and dispatching systems for commercial operations have been developed. In Chile, for example, a computerized system called ASICAM has been in use since 1990. It uses simulation with an embedded heuristic to produce a complete trucking schedule for one day of operations for more than 100 trucks (Epstein et al., 2007). The implementation of ASICAM in real operations has led to reductions in costs between 10% and 20% (Weintraub et al., 1996). Similar systems can be found in other countries. In Finland, a system called EPO/KUORMA was developed to deal with all stages of planning, from strategic to operation (Savola 2004, Palmgren et al., 2004). In Sweden, Skogforsk developed a system called FlowOpt (Forsberg et al., 2005), which integrates Geospatial Information Systems (GIS) with a database and uses a heuristic approach based on a Tabu Search algorithm. In Canada, a system named MaxTour was developed to compute the potential in back-haulage tours within the volume of one or several types of products, based on an adaptation of the well-known savings heuristic of Clarke and Wright (Marier et al., 2007).
In Australia, a SA-based truck scheduling system called FastTRUCK has been
developed for in-field chipping operations and log transport (Acuna et al.,
2011, Acuna et al., 2012). Worldwide, these decision support systems have
been tested by a number of forest companies, and potential savings in transport
costs between 2% and 10% have been reported.

This present paper is organized as follows: In Section 2, we present the
trick scheduling problem and introduce a small example. In Section 3 we
present a model formulation. Our SA-based solution approach, including
parameters and improvement methods, is presented in Section 4, while the
results of the numerical experiments are provided in Section 5. Finally, our
conclusions are presented Section 6, including recommendations for
implementing trucks scheduling systems in real operations, as well as future
research work.

2. PROBLEM DESCRIPTION

The problem to be solved considers a heterogeneous fleet of trucks (T)
that need to be optimally allocated to different transport tasks on a daily basis.
The main objective of this optimal truck allocation is to minimize total
unloaded travel times (TU) and waiting times (WT) at delivery points (wood
pickup locations), assuming that a central transportation system provides
schedules for trucks and ensures that their utilization is maximized.

In order to provide the inputs to solve the truck scheduling mathematical
model, the daily transport tasks must be predefined with the assistance of a
wood flow optimization model (not presented in this paper) which in turn is
part of a short term harvest model. A task corresponds to a truck load or
transport order that must be performed to obtain a feasible solution. As pointed
out by Oberscheider et al. (2013), the problem structure with tasks occurring at
depots and customers, reduces the number of constraints and variables
considerably, as opposed to vehicle routing problems without predefined
transport tasks.

The following attributes associated with transport tasks, trucks, and tours
are considered in the problem:
Tasks

- A transport task starts at a pickup location and ends at a customer site (mill).
- Each transport task has to be fulfilled.
- Each task is characterized by a wood pickup location (forest coupe or log landing), destination (mill), product, loading time (L), travel loaded (LT), unloading time (U), and time window at the pickup location [a,b].
- A number of tasks can only be performed by certain trucks. Some trucks have limited or no access to certain wood pickup points, or they are not able to carry some wood products (logs of different specifications).

Trucks

- A heterogeneous fleet of logging trucks of the same capacity is assumed.
- Trucks are based at different depots (D). One specific depot can be the daily starting point of one or several trucks.
- Trucks are unloaded at the customer sites based on first in first out servicing. A truck follows a queue if there is one at the wood pickup point. In order to maximize truck utilization, waiting times of trucks in queue is not wanted and must be minimized or eliminated completely.
- Each truck has a maximum shift time that must be met. The shift time includes travel empty, travel loaded, waiting and loading times at pickup points, and unloading times at customers.

Tours

- A truck always starts its daily tour empty at the corresponding depot.
- After leaving the depot, a truck visits a wood pickup location where is fully loaded. Neither partial loads nor transport between different wood pickup points is allowed.
A Simulated Annealing Algorithm to Solve the Log-Truck …

- After delivering and unloading the load at the customer, the truck can continue the tour by performing another transport task (provided there are remaining loads to be carried from that particular wood pickup point to any mill) or returning to its depot.
- During one daily tour a truck can visit one or more wood pickup points or customers, which in turn, can be visited by several trucks throughout the day.
- A tour must satisfy constraints in terms of the shift time (SHT) and time windows at depots and pickup points.

Figure 1. Small truck scheduling problem.

Table 1. Description of transport tasks for the small truck scheduling problem shown in Figure 1

<table>
<thead>
<tr>
<th>Task</th>
<th>Wood pickup location [W]</th>
<th>Customer site [C]</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>7 &amp; 8</td>
<td>7</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>7</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Complimentary Contributor Copy
The problem described above is illustrated with a small example (Figure 1), which includes 11 transport tasks, 2 truck depots, 7 wood pickup points, 3 customer sites, and two products. Details of the tasks including wood pickup point, customer site and product are presented in Table 1. It is worth noting that two or more transport tasks can share the same wood pick point (e.g. tasks 2 and 11), the same customer site (e.g. tasks 9, 10, and 11), or the same wood pickup point and customer site (tasks 7 and 8). Also, a customer can demand just a single product (e.g. customers 1 and 2), or two different products (e.g. customer 3). Each product is defined by a different log length combination.

2. MATHEMATICAL MODEL

The mathematical model corresponds to an adaptation of the formulation presented by Gronalt and Hirsch (2007) and Oberscheider et al. (2013), and it is formulated as a standard MDVRPPDTW problem. The model optimally allocates a heterogeneous fleet of logging trucks to transport tasks assuming a centralized transport system is in place to provide schedules for trucks and ensure that their utilization is maximized.

Two sets of binary variables \( X_{ijt} \), \( Y_{it} \), and two set of continuous variables \( A_{it} \), \( WT_i \) are defined:

\[
X_{ijt} = \begin{cases} 
1, & \text{if truck } t \text{ performs task } j \text{ after task } i \\
0, & \text{otherwise}
\end{cases}
\]

\[
Y_{it} = \begin{cases} 
1, & \text{if truck } t \text{ performs task } i \\
0, & \text{otherwise}
\end{cases}
\]

\[
A_{it} = \text{Arrival time of truck } t \text{ to harvest site for task } i
\]

\[
WT_i = \text{Waiting time at harvest site for task } i
\]

In addition, a set \( L \) of depot (D) and transport (E) tasks is defined, with \( L = D \cup E \).

\[
\text{Min} \sum_{i \in L} \sum_{j \in L} \sum_{t \in T} (TU_{ij} + WT_i) \cdot X_{ijt} \quad (1)
\]
subject to:

\[
\sum_{i \in L} X_{iht} - \sum_{j \in L} X_{hjt} = 0 \forall h \in L, t \in T
\]  \tag{2}

\[
\sum_{j \in L} \sum_{r \in T} X_{ijt} = 1 \forall i \in L
\]  \tag{3}

\[
\sum_{j \in L} X_{ijt} + X_{ilt} = 1 \forall i \in D, t \in T \ (i = t)
\]  \tag{4}

\[
\sum_{i \in L} X_{ijt} + X_{fjt} = 1 \forall j \in D, t \in T \ (j = t)
\]  \tag{5}

\[
Y_{it} = \sum_{j \in L} X_{ijt} \forall i \in L, t \in T
\]  \tag{6}

\[
\sum_{ij} T U_{ij} \cdot X_{ijt} + \sum_{i} (L_i + W T_i + T L_i + U_i) \cdot Y_{it} \leq S H T \forall t \in T
\]  \tag{7}

\[
A_{it} + W T_i + L_i + T L_i + U_i + T U_{ij} - M (1 - X_{ijt}) \leq A_{jt} \forall i, j \in E, t \in T
\]  \tag{8}

\[
A_{it} + W T_i \geq a_i \forall i \in E, t \in T
\]  \tag{9}

\[
A_{it} \leq b_i \forall i \in E, t \in T
\]  \tag{10}

\[
X_{ijt} \in \{0, 1\} \forall i \in L, j \in L, t \in T
\]  \tag{11}
The objective function (1) minimizes driving times of empty trips and waiting times at wood pickup points. Constraint (2) forces a truck to do an additional task after completing the previous one. Constraint (3) ensures that every task (from depot and transport tasks) is performed. Constraint (4) specifies that at the beginning of the day, each truck may perform a transport task or remains at the depot. Constraint (5) specifies that a truck may perform a new transport task after completing the previous one, or return to the depot. Constraint (6) links the binary variables. Constraint (7) controls that shift times of drivers are not exceeded. Constraints (8) to (10) guarantee that time windows at wood pickup points are met. The parameter M in constraint (8) is used to linearize the equation. Constraints (11) and (12) ensure the binarity of the decision variables $X_{ijt}$ and $Y_{it}$. Constraints (13) and (14) establish the non-negativity of variables $A_i$ and $WT_i$.

3. Simulated Annealing Algorithm for the Truck Scheduling Problem

Our solution approach to solve the truck scheduling problem is based on a standard SA procedure (Figure 2), and it was implemented and programmed in Visual C++ using an object-oriented design. It includes the SA heuristics as an optimization engine in combination with a deterministic discrete event simulation to emulate the movement of trucks throughout the day. In addition the simulation model keep tracks of all the performance metrics associated with the daily tasks performed by the trucks, and provide general metrics for the whole fleet.
Our solution approach consists of the following steps:

1. Determination of the SA cooling schedule, including parameters testing (section 3.1)
2. Construction of a initial feasible solution with an greedy heuristic (section 3.2)
3. Solution improvement by applying four different methods (section 3.3)

At the beginning, all the SA parameters associated with the cooling scheme (initial temperature, temperature reduction factor, iterations per temperature and final temperature) are set up.
In successive iterations, the next solution $X_t'$ is generated from $N(X_t)$ by randomly applying one of the solution improvement methods, and its objective function $S_t$ is evaluated accordingly. If the temporary objective function $S_t$ is lower than the current objective function $S_c$, then the move is accepted and the new current objective function $S_c$ and solution $X_c$ are set equal to the current temporary objective function $S_t$ and temporary solution $X_t$, respectively. If the temporary objective function $S_t$ is greater than the current objective function $S_c$, the probability of accepting this worse solution (non-improving move) is determined by the Boltzmann probability function $P = \exp(-\Delta/T)$, where $\Delta$ is the difference between the temporary $S_t$ and current $S_c$ solutions, and $T$ is the current temperature. If the value of the function $P$ is greater than a random variable between 0 and 1, then the move is accepted and the temporary objective function $S_t$ and solution $X_t$ become the new current objective function $S_c$ and solution $X_c$. Subsequently, the current objective function $S_c$ is compared to the current best objective function $S_b$. If $S_c$ is lower than $S_b$, then $S_c$ becomes the new best solution $S_b$, and the current solution $X_t$ becomes the new best solution $X_b$. This search continues for a number of iterations at each temperature, after which the temperature is lowered by applying a reduction factor. The algorithm is stopped if the current temperature $T$ is lower than the final temperature parameter final_temp.

3.1. Determination of the Cooling Schedule

Romeo and Sangiovanni-Vincentelli (1991) state that an effective cooling schedule is essential to reducing the amount of time required to find an optimal solution. Therefore much of the literature on cooling schedules (Nourani and Andersen 1998, Cohn and Fielding 1999) has been devoted to this topic.

The cooling schedule of the simulated annealing algorithm implemented is fully defined by the following components: Initial temperature ($T_0$), final temperature (FT), temperature reduction factor ($\alpha$), and Iterations at each temperature (niter). In our SA algorithm, this cooling scheme was implemented statically, with all the components being completely specified before the algorithm began (Henderson et al., 2003).

**Initial temperature**

Several tests were conducted to define the initial temperature $T_0$. The theory establishes that the initial temperature must be hot enough to allow
moves to almost every neighborhood state. Starting at too low an initial
temperature may result in hill climbing and stalling at a local minimum.
Starting at a very high temperature will increase the period of time in which
the random search is conducted (Henderson et al., 2003). Basically, the SA
model was run several times to determine the maximum change in the
objective function. In our algorithm, we evaluated the use of three different
starting temperatures. This was done by setting up a high initial temperature,
which was decreased from 100% until 80% and 60% of non-improving moves
were accepted (the last alternative has been suggested by Rayward-Smith et al.
(1996)). Because many non-improving moves are accepted early in the
solution procedure, the algorithm does not depend on a good initial solution.

**Final temperature**

It is usual to select an ending temperature where the probability of
accepting a worse solution is very low. Letting the temperature decrease until
it reaches to zero can make the algorithm run for a lot longer, especially when
a geometric cooling schedule with a very low temperature reduction factor is
being used (as in this case). In practice, it is not necessary to let the
temperature reach zero, as temperatures close to zero avoid accepting worse
solutions. Therefore, a suitable low temperature was set as the final
temperature (FT), in order to maintain a low chance (less than 1%) of
accepting worse solutions.

**Temperature reduction factor**

The temperature reduction factor $\alpha$ is critical to the success of the
algorithm. Stenski and Kirkpatrick (1991) presented an exact (non-heuristic)
characterization of annealing schedules. Their experiments suggest that
optimal cooling schedules are not monotonic decreasing in temperature. They
have also shown that for some problems geometric (of the form $T = \alpha \cdot T$ as
implemented in our algorithm) and linear cooling schedules perform better
than inverse logarithmic cooling schedules, when sufficient computing effort
is allowed. They also observed that geometric cooling schedules are not
greatly affected by excessively high initial temperatures.

In practical terms, there is a compromise between the number of iterations
at each temperature required to stabilize the system at that temperature and
their exponential increase with problem size. This can be implemented by
allowing the algorithm to run a large number of iterations at a few
temperatures, a small number of iterations at many temperatures or a balance
between the two. Experience in previous truck scheduling problems solved
with SA (Haridass 2009, Lin et al., 2009) has shown that $\alpha$ should be between 0.9 and 0.99, with better results being found in the higher end of the range. Of course, the higher the value of $\alpha$, the longer it will take to decrement the temperature and reach the stopping criterion.

**Iterations at each temperature**

The number of iterations (commonly a constant number) at each temperature determines the computational effort required to run a particular cooling schedule, and it is determined by the size of the neighbourhood and solution space (Dowsland 1993). The value sets how many improvements changes (truck swaps and insertions) will be made before a reduction of the temperature. As pointed out previously, the theory behind the algorithm states that before a reduction of the temperature, a sufficient number of iterations should take place so that a state of equilibrium is reached. An alternative is to dynamically change the number of iterations as the algorithm progresses. At lower temperatures it is important that a large number of iterations are done so that the local optimum can be fully explored. At higher temperatures, the number of iterations can be less.

### 3.2. Construction of an Initial Feasible Solution

The solution approach commences with the construction of an initial feasible solution through a greedy heuristic. This initial solution must not violate any of the constraints associated with trucks and tours. The procedure of the heuristics is as follows:

1. Selection of a truck $t \in T$ randomly and associated depot task $i \in D$.
2. Allocation of selected truck and associated depot task to the closest feasible transport task $i \in E$. This step guarantees that a truck is only allocated to a task whose wood pickup and destination points are accessible by the truck, and whose product can be transported by the truck.
3. Continue with steps 1) and 2) until a feasible tour is completed for the truck selected
4. Continue with the rest of the trucks and their associated depot tasks

Given the random selection of trucks in step 1), the initial solution constructed in this heuristic procedure is also random and varies in each run of
the simulated annealing algorithm. A feasible solution for a truck can be represented as a list of tasks performed on a tour. Figure 3 shows a feasible solution representation for the small problem presented in Figure 1. This includes 4 trucks, 11 transport tasks, 7 wood pickup points, and 3 customers. Each truck departs from its own depot and performs a number of tasks during its daily tour (3 tasks by trucks 1, 2, and 3, and 2 tasks by truck 4). The duration between one transport task and the next one is given by the empty travel time that takes for a truck to go from a customer site to the next wood pickup location.

3.3. Improvement Heuristics to Explore Neighborhood

If the available computing time is to be used efficiently, it is important that the SA implementation routines are as fast as possible. This means that the solution space should be kept as small as possible (a challenge in most combinatorial problems where infeasible solutions are included during the search of good solutions). In addition to keeping the solution space small, it is also useful to aim for reasonably small neighborhoods during the search, enabling a neighborhood to be searched adequately in fewer iterations, although there is less opportunity for dramatic improvements to occur in a single move (Dowsland 1993).

The neighborhood structure implemented in our SA algorithm includes the following improvement methods.

1. Single task insertions
2. Single task swaps
3. Task insertions for a set of n-trials
4. Task swaps for a set of n-trials

Each method is selected and applied with a certain probability to generate candidate routes for randomly selected trucks. A set \( N(X_i) \) is defined to be the set of solutions neighboring a solution \( X_i \). In each iteration a new solution \( X_i' \) is generated from \( N(X_i) \) by using some of the four improvement methods. Because the use of computing resources is more intensive with insertions and swaps between trucks for a set of n-trials, each of these two improvements methods was selected with a maximum probability of 20%, while the standard insertion and swap methods were selected with a maximum probability of 60% and 80%, respectively.
The insertion method is used to insert the task from one truck’s route in the route of another truck. The main effect of the insertion method is the reduction of the total unloaded distance travelled by the trucks as well as a reduction of fleet size. Two trips and two trucks are selected randomly from \( N(X_t) \). The task from the first truck’s route is inserted in the position right before the random task selected from the second truck’s route. Prior to applying the insertion, it is necessary to check if the truck being inserted a new task can actually perform that task (the truck cannot have access to a wood pickup location or is not able to carry the product associated with the task). Also, it is necessary to check if the route with the new task inserted exceeds the daily shift time of the truck driver.

Figure 4 shows a graphical representation before (a) and after (b) the insertion method is applied to trucks 2 and 4 in the small example presented in Figure 1 and initial solution presented in Figure 3.

Before the insertion method is applied, the route of truck 2 consists of transport tasks 8, 5, and 1, while the route of truck 4 consists of transport tasks 11 and 3. Then, task 1 from truck 2’s route is randomly selected and inserted after task 11 in truck 4’s route. Consequently, the new route for truck 2 now consists of transport tasks 8 and 5, while the new route for truck 4 now consists of tasks 11, 1, and 3.

The swap method is used to exchange the task from one truck’s route with the task from a second truck’s route. The swap is performed by randomly selecting two trucks and swapping a randomly selected single load pair between the two different routes. This method is often used to improve feasible solutions when any single load insertion is not applicable in the current solution. Like the insertion method, prior to applying swap, it is necessary to check out if the truck being inserted a new task can actually...
perform that task and if the route with the new task inserted exceeds the daily shift time of the truck driver.

Figure 4. Graphical representation before (a) and after (b) the insertion method is applied to trucks 2 and 4 in the small truck scheduling example.

Figure 5 shows a graphical representation before (a) and after (b) the swap method is applied to trucks 2 and 4 in the small example presented in Figure 1 and initial solution presented in Figure 3. Again, before the swap method is applied, the route of truck 2 consists of transport tasks 8, 5, and 1, while the route of truck 4 consists of transport tasks 11 and 3. Then, task 1 from truck 2’s route is randomly selected and swapped with task 3 from truck 4’s route which is also selected randomly. After the swap method is applied, the new route for truck 2 now consists of transport tasks 8 and 5, while the new route for truck 4 now consists of tasks 11, 1, and 3.
Complementary, with the aim to increase the chance of obtaining a better solution, insertion and swaps between a pair of trucks were performed for a set of n-trials (as implemented by Li et al., 2009). The number of trials n was established by selecting 20% of the fleet in the current temporary solution Xt, and the best solution of the trial was selected as the new solution (Xt’).

4. COMPUTATIONAL RESULTS

4.1. Problem Instances

Truck scheduling in real-life problems can involve a considerable number of trucks, transport tasks, customers and wood pickup locations. For example,
in Australia and New Zealand it is not uncommon that more than 100-150 transport tasks have to be performed on a daily basis to move logs from wood pickup locations (harvest sites) to customers. Smaller problems have been solved using different optimization and simulation techniques. Robinson (1995) presents a case study with 24 truck and 86 transport tasks in the North Island of New Zealand. Murphy (2003) presents another case study in the South Island on New Zealand, involving 9 trucks and 35 transport tasks per day. Walsh et al. (2013) presents two case studies in Australia: one with 31 trucks and approximately 93 daily transport tasks, and one with 16 trucks and about 45 daily transport tasks.

Two small sets of problem instances generated from real-life data were used to test and validate our algorithmic approach. Each instance consisted of a sub-set of 30 transport tasks and 10 trucks, taken from an original data set comprising more than 150 transport tasks, 60 trucks, 25 wood pickup locations, and 8 customers. In the first instance, trucks were randomly allocated to 3 depots located at an average distance of 50 km from wood pickup locations (range 31-57 km), while the distance from wood pickup locations to customer sites ranged from 37 to 174 km (average 87 km). In the second instance, all the trucks were allocated to central depot located at an average distance of 82 km from wood pickup locations (range 61-118 km). In both instances, the same wood pickup locations and customers were used for the analysis.

Our SA algorithmic approach was implemented and programmed with Visual C++, while the mathematical model formulation was implemented and solved with the software GAMS® v. 24.1.3 and the solver CPLEX®. Both the SA algorithm and the optimization model were run and solved with a computer equipped with an Intel® Core i7 processor (2.3 GHz) and 8 GB of RAM memory.

4.2. SA Cooling Schedule Testing

In each problem instance, a total of 36 SA cooling schemes were tested to analyze their effect on the quality of the solutions. These schemes included three initial temperatures (20000, 40000, and 100000), four temperature reduction factors (0.8, 0.95, 0.99, and 0.999), three sets of iterations per temperature (500, 1000, and 1500), and five runs per scheme. The initial temperatures were set in such a way that about 60%, 80%, and 100% non-improving moves were accepted at the beginning of the algorithm. In all the
Mauricio Acuna and John Sessions

experiments the final temperature was set to 1000 (with less than 1% of the non-improving solutions being accepted).

Table 2 and 3 show the percent deviation between the average solution (five runs) in each cooling scheme and the best single SA solution obtained in problem instances one and two, respectively. A total of 180 runs were made in each problem instance. The best single SA solution in problem instance 1 was obtained with an initial temperature of 20000, a temperature reduction factor of 0.999, and 1000 iterations per temperature, while the best single SA solution obtained in problem instance 2 was obtained with an initial temperature of 40000, a temperature reduction factor of 0.999, and 1500 iterations per temperature. The maximum deviations between the average solution and the best single solution were 3.6% and 5.1%, for problem instances 1 and 2, respectively.

A multi-way analysis of variance (ANOVA) was performed with the data set from each problem instance to determine the statistical significance of the differences between levels associated with each cooling scheme parameter (experimental factors). The ANOVA analysis indicated that the differences between levels were only significant for the SA parameters temperature adjustment factor and number of iterations per temperature (p-value < 0.05). The fact that there is not statistically significant difference between levels of the initial temperature factor (20000, 40000, and 100000) is in accordance with the results of previous studies (Lin et al., 2009, Haridass 2009). This suggests that the initial temperature of the SA algorithm should be set in such a way that only a proportion of the non-improving moves are accepted at the beginning of the algorithm, as opposed to accepting all non-improving moves early in the solution process.

Results from Tables 2 and 3 also show that the temperature reduction factor is the parameter with the biggest impact on the quality solutions of the SA algorithm. In comparison with the best single SA solution, average deviations drop as the temperature reduction factor rises from 0.8 to 0.999. For problem instance 1, average deviations were 3.1%, 2.4%, 1.8%, and 1.1%, for temperature reduction factors of 0.8, 0.95, 0.99, and 0.999, respectively, while for problem instance 2, average deviations were 3.9%, 2.9%, 2.1%, and 1.2%, for temperature reduction factors of 0.8, 0.95, 0.99, and 0.999, respectively.
Table 2. Percent deviation between the average solution in each cooling scheme and the best single SA solution obtained in problem instance 1

<table>
<thead>
<tr>
<th>Iterations per temperature</th>
<th>Temperature reduction factor</th>
<th>0.8</th>
<th>0.95</th>
<th>0.99</th>
<th>0.999</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Initial Temperature</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20000</td>
<td>0.8</td>
<td>5.0</td>
<td>3.5</td>
<td>3.6</td>
<td>2.7</td>
</tr>
<tr>
<td>40000</td>
<td></td>
<td>3.5</td>
<td>2.9</td>
<td>2.6</td>
<td>2.2</td>
</tr>
<tr>
<td>10000</td>
<td></td>
<td>3.6</td>
<td>2.9</td>
<td>2.6</td>
<td>2.2</td>
</tr>
<tr>
<td></td>
<td>Initial Temperature</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20000</td>
<td>0.95</td>
<td>3.2</td>
<td>2.8</td>
<td>3.0</td>
<td>2.7</td>
</tr>
<tr>
<td>40000</td>
<td></td>
<td>2.8</td>
<td>2.0</td>
<td>2.4</td>
<td>1.6</td>
</tr>
<tr>
<td>10000</td>
<td></td>
<td>2.8</td>
<td>2.0</td>
<td>2.4</td>
<td>1.6</td>
</tr>
<tr>
<td></td>
<td>Initial temperature</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20000</td>
<td>0.99</td>
<td>2.7</td>
<td>2.4</td>
<td>2.1</td>
<td>1.6</td>
</tr>
<tr>
<td>40000</td>
<td></td>
<td>2.4</td>
<td>2.2</td>
<td>2.1</td>
<td>1.6</td>
</tr>
<tr>
<td>10000</td>
<td></td>
<td>2.4</td>
<td>2.2</td>
<td>2.1</td>
<td>1.6</td>
</tr>
</tbody>
</table>

Table 3. Percent deviation between the average solution in each cooling scheme and the best single SA solution obtained in problem instance 2

<table>
<thead>
<tr>
<th>Iterations per temperature</th>
<th>Temperature reduction factor</th>
<th>0.8</th>
<th>0.95</th>
<th>0.99</th>
<th>0.999</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Initial Temperature</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20000</td>
<td>0.8</td>
<td>5.1</td>
<td>4.2</td>
<td>4.3</td>
<td>3.1</td>
</tr>
<tr>
<td>40000</td>
<td></td>
<td>4.2</td>
<td>3.1</td>
<td>3.0</td>
<td>2.7</td>
</tr>
<tr>
<td>10000</td>
<td></td>
<td>4.3</td>
<td>3.1</td>
<td>3.0</td>
<td>2.7</td>
</tr>
<tr>
<td></td>
<td>Initial Temperature</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20000</td>
<td>0.95</td>
<td>4.0</td>
<td>3.2</td>
<td>3.5</td>
<td>2.8</td>
</tr>
<tr>
<td>40000</td>
<td></td>
<td>3.2</td>
<td>3.1</td>
<td>2.6</td>
<td>2.2</td>
</tr>
<tr>
<td>10000</td>
<td></td>
<td>3.5</td>
<td>3.1</td>
<td>2.6</td>
<td>2.2</td>
</tr>
<tr>
<td></td>
<td>Initial temperature</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20000</td>
<td>0.99</td>
<td>3.5</td>
<td>3.9</td>
<td>3.5</td>
<td>2.8</td>
</tr>
<tr>
<td>40000</td>
<td></td>
<td>3.9</td>
<td>3.9</td>
<td>3.5</td>
<td>2.8</td>
</tr>
<tr>
<td>10000</td>
<td></td>
<td>3.5</td>
<td>3.9</td>
<td>3.5</td>
<td>2.8</td>
</tr>
</tbody>
</table>

Boxplots in Figures 6 and 7 show the deviations between each of the 180 runs (grouped by temperature reduction factor) and the best single SA solution in problem instances 1 and 2, respectively. Deviations present a higher variation with reduced temperature reduction factors. In the case of the problem instance 1, the standard deviations are 0.66, 0.59, 0.41, and 0.37, for temperature reduction factors of 0.8, 0.95, 0.99, and 0.999, respectively. For problem instance 2, the standard deviations are 0.99, 0.60, 0.59, and 0.40, for temperature reduction factors of 0.8, 0.95, 0.99, and 0.999, respectively.

These results show that the best solutions are obtained when the temperature is reduced very slowly (reduction factor of 0.999) in combination with reduced initial temperatures (20000 or 40000) and number of iterations per temperature (500 or 1000). It is also clear that the convergence of the
algorithm is improved when a slow cooling scheme is implemented (slower temperature reduction factor), which results in more stable solutions and lower variance of the deviations in relation to the best single SA solution.

Figure 6. Boxplot depicting deviations between single and best SA solutions in problem instance 1.

Figure 7. Boxplot depicting deviations between single and best SA solutions in problem instance 2.
4.3. Comparison of Different Neighborhood Structures

Using data from problem instance 1, five different neighborhood structures were analyzed to determine their impact on the quality of the solutions in relation to the best single SA solution (Table 3). Each neighborhood structure implemented in each iteration of the SA algorithm is identified by a 4-code system. The codes (first column in Table 3) represent the probability of an insertion move, the probability of a swap move, the probability of an insertion move for a set of n-trials, and the probability of a swap move for a set of n-trials, respectively. In each neighborhood structure the four code probabilities add up 100%. When a set of n-trials was used, the number n corresponded to the transport tasks of approximately 30% of the truck fleet, all of which were randomly selected for insertions and swaps. The best solution obtained for the set of trials was used as a temporary solution for the current iteration.

<table>
<thead>
<tr>
<th>Neighborhood structure</th>
<th>Average SA solution (min)</th>
<th>Best SA solution (min)</th>
<th>Deviation (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>60 / 20 / 10 / 10</td>
<td>2685</td>
<td>2660</td>
<td>0.0</td>
</tr>
<tr>
<td>80 / 0 / 20 / 0</td>
<td>2697</td>
<td>2673</td>
<td>0.5</td>
</tr>
<tr>
<td>40 / 40 / 10 / 10</td>
<td>2700</td>
<td>2687</td>
<td>1.0</td>
</tr>
<tr>
<td>20 / 60 / 10 / 10</td>
<td>2706</td>
<td>2693</td>
<td>1.2</td>
</tr>
<tr>
<td>0 / 80 / 0 / 20</td>
<td>2841</td>
<td>2805</td>
<td>5.4</td>
</tr>
</tbody>
</table>

For the analysis of the neighborhood structures, the SA parameters were specified based on the results of the cooling scheme testing, with an initial temperature of 20000, a temperature reduction factor of 0.999, and 1000 iterations per temperature.

Five runs of the SA algorithm were performed within each neighborhood scheme, and the average solutions were compared with the best single SA solution. The latter was obtained with a 60/20/10/10 neighborhood structure and used as the target value (2660 min) for the comparisons.

Solutions were quite poor for neighborhood structures that accepted swap moves with a high probability. Conversely, solutions improved to some point when a neighborhood structure that accepts insertion moves with a high probability was used.
However, the best results were obtained for a neighborhood structure that combines a high probability of insertion moves, a low probability of swap moves, and a reduced probability of both insertion and swaps (10% each) for a set of n-trials. In fact, all the neighborhood structures that combine insertion and swaps moves with some probability result in deviations of less than 1.2% in comparison to the best single SA solution. This seems to be the result of a better exploration of the space of feasible solutions when combined insertion and swap moves are selected with some probability.

4.4. Comparison of SA and Optimal Solutions

For big problem instances, the computational complexity of the truck scheduling problem precludes the use of traditional optimization techniques and standard solver software for its solution. However, for smaller problem instances (such as the ones used in this paper), the mathematical model can still be solved with standard optimization techniques and solvers, such as the branch and bound algorithm, to solve discrete mixed-integer linear problems.

In order to compare the efficiency of our SA heuristics in problem instances 1 and 2, we compared the best SA solution obtained in each temperature adjustment factor group with the optimal solution obtained with the optimization software GAMS and the solver CPLEX. The mathematical formulation of the problem solved with an optimization solver consisted of more than 16,000 integer variables and 17,000 constraints.

Table 5 and 6 shows the results for problem instances 1 and 2, respectively. In all the cases, a 60/20/10/10 neighborhood structure was used to perform the comparisons. In problem instance 1, the best SA solution (2660 min) was obtained with a temperature factor of 0.999, an initial temperature of 20000, and 1000 iterations per temperature.

This solution is 0.4% away from the optimal solution (2648 min) obtained with GAMS® and CPLEX®. In the case of problem instance 2, the best SA solution (2689 min) was obtained with a temperature factor of 0.999, an initial temperature of 40000, and 1500 iterations per temperature. This solution is 0.7% away from the optimal solution (2670 min) obtained with GAMS® and CPLEX®.
Table 5. Comparison between SA and optimal solutions for problem instance 1. MIP solution is 2648 minutes

<table>
<thead>
<tr>
<th>Parameters SA</th>
<th>SA solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temperature adjustment factor</td>
<td>Initial temperature</td>
</tr>
<tr>
<td>0.8</td>
<td>20000</td>
</tr>
<tr>
<td>0.95</td>
<td>40000</td>
</tr>
<tr>
<td>0.99</td>
<td>40000</td>
</tr>
<tr>
<td>0.999</td>
<td>20000</td>
</tr>
</tbody>
</table>

Table 6. Comparison between SA and optimal solutions for problem instance 2. MIP solution is 2670

<table>
<thead>
<tr>
<th>Parameters SA</th>
<th>SA solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Temperature adjustment factor</td>
<td>Initial temperature</td>
</tr>
<tr>
<td>0.8</td>
<td>20000</td>
</tr>
<tr>
<td>0.95</td>
<td>20000</td>
</tr>
<tr>
<td>0.99</td>
<td>40000</td>
</tr>
<tr>
<td>0.999</td>
<td>40000</td>
</tr>
</tbody>
</table>

The results show that considering all the scenarios, the best SA solutions were less than 3% away from the optimal solution. In both problem instances, the best solutions were obtained with very slow temperature reduction factors, a reduced initial temperature and a reduced number of iterations per temperature.

**CONCLUSION**

In this paper, we have presented a simulated annealing algorithm to solve the log truck scheduling problem, which is a variation of the timber truck vehicle routing problem with time windows (TTVRPTW). With forestry industry transportation costs running at million dollars per day, transport planning is an area in need of optimization to realize cost savings. More efficient truck scheduling algorithms and software tools must be developed and implemented so they can assist the forestry industry to reduce these costs.

Complimentary Contributor Copy
by demonstrating the benefits of investing in transport planning systems and encouraging forestry companies to move to centralized planned transport systems.

The mathematical formulation of the truck scheduling problem proposed in the paper is an adaptation of the model presented by Gronalt and Hirsch (2007) and Oberscheider et al. (2013). The model optimally allocates a heterogenous fleet of logging trucks to transport tasks assuming a centralized transport system is in place to provide schedules for trucks and ensure that their utilization is maximized. The problem structure with tasks occurring at depots and customers reduces the number of constraints and variables considerably, as opposed to vehicle routing problems without predefined transport tasks.

Given the computational complexity of the truck scheduling problem, a SA heuristic has been developed, implemented, and tested for two small problem instances, each consisting of 30 transport tasks and 10 trucks. Our results show that the best solutions are obtained when the temperature is reduced very slowly (reduction factor of 0.999) in combination with reduced initial temperatures (20000 or 40000) and number of iterations per temperature (500 or 1000). In addition, the convergence of the algorithm is improved when a slow cooling scheme is implemented (slower temperature reduction factor), which results in more stable solutions and lower variance of the deviations in relation to the best single SA solution.

This cooling scheme in combination with a neighborhood structure that includes a high probability of insertion moves, a low probability of swap moves, and a reduced probability of both insertion and swaps (10% each) for a set of n-trials, resulted in the best solutions with our SA algorithm. This seems to be the result of a better exploration of the space of feasible solutions when combined insertion and swap moves are selected with some probability. In comparison to the optimal solution obtained with a software solver, the best SA solutions were less than 3% away from the optimal solution. In both problem instances, the best solutions were obtained with very slow temperature reduction factors, a reduced initial temperature and a reduced number of iterations per temperature. The results from all these numerical experiments show that our SA algorithm is able to solve real-life problems with high quality solutions.

Future research projects should focus on testing the solution approach and its efficiency with bigger problem instances, so that the algorithm and software tool can be implemented in real operations by transport planners and truck dispatchers. In addition, other operational constraints should be added to the
A Simulated Annealing Algorithm to Solve the Log-Truck ...

algorithm to better capture operational constraints of the real-life problems, such as driving restrictions at peak traffic hours, overnight on-carts (leaving trucks loaded at the depots), and fatigue management rules. Finally, more research should be done to provide innovative computational technology and integrated schedule and real-time dispatching support systems to forest companies and hauling contractors, so that their transportation activities result in improved efficiency, effectiveness and safety.

ACKNOWLEDGMENT

This research was partially funded by the Northwest Renewables Alliance (NARA). NARA is supported by the Agriculture and Food Research Initiative Competitive Grant 2011-68005-30416 from the US Department of Agriculture, National Institute of Food and Agriculture.

REFERENCES


A Simulated Annealing Algorithm to Solve the Log-Truck … 219

107-113.


Complimentary Contributor Copy
Mauricio Acuna and John Sessions


Chapter 9

**Potentially Applications of Simulated Annealing to Multi-Species Fish Habitat Conservation in the Gulf of Maine**

*Rosamonde R. Cook*¹,* and *Peter J. Auster*¹,²

¹Northeast Underwater Research, Technology & Education Center, University of Connecticut at Avery Point, Groton, CT, US
²Sea Research Foundation - Mystic Aquarium, Mystic, CT, US

**Abstract**

Simulated annealing has been used successfully in spatial land use planning and nature reserve system design in many parts of the world for more than a decade. While the methods are similar, the underlying data vary in kind and quality. In this chapter, we use standardized fisheries independent survey data from a long term monitoring program to demonstrate the effectiveness of simulated annealing for identifying efficient design scenarios for marine protected areas intended to satisfy the Essential Fish Habitat requirements of the U.S. Sustainable Fisheries Act for 15 economically valuable demersal fish species in the Gulf of Maine on the U.S. Eastern Continental Shelf. Results that ensure

---

* Corresponding author: Rosamonde Cook, Western Riverside County Biological Monitoring Program, 4500 Glenwood Drive Bldg. C, Riverside, California, 92501. Phone: 951-320-2168. Fax: 951-782-4847. Email: RCOok@biomonitoringrca.org.
minimum total area requirements for conservation targets stated in terms of percent of total regional abundance of individual species from research trawl surveys achieve the goal of capturing those locations where most species occur at their highest region-wide abundances, and by inference, their highest quality habitats, under a range of spatial design scenarios. Significantly, these scenarios also represent the diversity of the wider demersal fish community and capture some of the highest local densities of many species not targeted in fishery operations, and would therefore help ensure the conservation of marine fish biodiversity overall. Using this approach can aid in developing options to solve a range of important and controversial management issues, described in this chapter. We conclude that simulated annealing is a valuable and vital tool for marine fisheries management on the Eastern Continental Shelf of the U.S., and likely in other parts of the world where systematic time series survey data exist.

**Keywords:** fish, habitat, simulated annealing, marine protected area, biodiversity

**INTRODUCTION**

Marine Protected Areas (MPAs) are defined areas of ocean where natural and/or cultural resources are given greater protection than the surrounding waters. Many scientists have concluded between 20% and 50% of the world’s oceans will need to be protected from fishing in order to significantly reduce the risks of overexploitation and fishery collapse, maintain genetic variation, optimize connectivity, and reduce by-catch (Roberts and Hawkins 2000). A call to achieve 20% protection by the year 2020 was made in a consensus statement (Troubled Waters: A Call for Action) signed in 1998 by more than 1600 scientists worldwide. Currently, however, less than 1% of the sea is fully protected. Given the total area in reserve is likely to be small, at least in the near term, it is important that reserves designed to preserve fisheries also protect biodiversity (Sala et al., 2002).

Design considerations for MPAs meant to enhance fisheries include size, number, shape, and spatial arrangement (Hastings and Botsford 2003). For example, large protected areas may be favored over several smaller ones (Fogarty 1999, Dayton et al., 2000) because the smaller boundary-area ratio will tend to reduce movement rates of fishes from inside to out (Polacheck 1990, Lindholm et al., 2001). Rounder, compact shapes may be preferred over long, thinner ones for the same reason. Smaller total boundary lengths may also reduce social and economic impacts of habitat protection as well as the
difficulty and expense of enforcement. However, multiple reserves may be necessary to protect multiple life stages and seasonal movement patterns.

To ensure protection of biodiversity, a system of MPAs would ideally be: 1) comprehensive (include the full range of biodiversity in a region), 2) adequate (contain sufficient numbers of individuals to ensure species remain viable in the long term), and 3) representative (sample species throughout their geographic range and variety of ecological communities) (Fernandes et al., 2005). Design considerations similar to those for fisheries enhancement are relevant here too. In theory, reserve networks can be designed to both enhance fisheries and maintain biodiversity (Sala et al., 2002, Hastings and Botsford 2003) however few studies have addressed this question with empirical data. In an exception involving near shore rocky reef fishes in southeastern Australia, Gladstone (2007) found that sites selected for higher densities of exploited fish species performed no better at capturing the diversity of non-exploited species than sites selected at random.

Simulated annealing (Kirkpatrick et al., 1983) and other kinds of complementarity-based reserve design algorithms (e.g., Pressey et al., 1993) have been developed for conservation planning in terrestrial systems to determine the minimum area required to achieve a particular conservation goal, such as inclusion of each species in at least one protected area or a proportion of each species range within a system of protected areas (Margules and Pressey 2000). Only recently have these algorithms been used in MPA planning (Sala et al., 2002, Game et al., 2008, Becker et al., 2009) and to test theory related to the conservation of biodiversity such as whether one group of species would serve as a good surrogate for the distribution or abundance of other taxa (e.g. Ward et al., 1999, Gladstone 2002, Beger et al., 2007).

Of the various reserve design algorithms available today, the program Marxan (Ball et al., 2009) is the most widely used around the world today (http://www.uq.edu.au/marxan/). Marxan uses simulated annealing to compare a large number of different combinations of spatial planning units in order to find combinations that meet specific conservation goals at minimum cost. Cost is defined mathematically by an objective function that includes the summed costs of individual planning units and penalties for failing to meet defined conservation goals.

\[
\text{Total Cost} = \sum_{\text{PU's}} \text{PU costs} + \text{BLM} \sum_{\text{PU's}} \text{boundary lengths} + \sum_{\text{species}} \text{SPF} \times \text{penalty} + \text{Threshold Penalty},
\]
where PU refers to planning units and SPF is the “Species Penalty Factor”, a term that weights the penalty of failing to meet the conservation target levels specified. BLM is the Boundary Length Modifier, a weighting that can be used to control the spatial aggregation of planning units. Increasing the BLM favors solutions that aggregate planning units into fewer, more compact protected areas. The threshold penalty is a cost applied to exceeding some maximum desired number of planning units (total area in reserve) and can be used to control the size of solutions. Setting a priori targets for the number and boundary length of areas may be desired to meet predetermined design specifications. An important aspect of the algorithm is that solutions will tend to include planning units that make the greatest contribution toward meeting the conservation goals because their selection maximizes cost efficiency.

**ESSENTIAL FISH HABITAT**

An important step toward greater conservation of marine fish habitats in the United States was passage of the Sustainable Fisheries Act in 1996 which requires designation of essential fish habitat (EFH), defined as “those waters and substrate necessary to fish for spawning, breeding, feeding, or growth to maturity” and regulations to minimize adverse effects on habitat by fisheries (Schmitten 1999). EFH, if managed to conserve fish populations, is consistent with the IUCN definition of a marine protected area (MPA) (Kelleher and Kenchington 1992). EFH must be defined for all U.S. federally managed species, i.e., species of economic importance, but no such requirements currently exist for the larger number of species not directly targeted in fishing operations but often taken as bycatch. Many are likely to play a significant functional role in marine ecosystems and are equally vulnerable to disturbance of their benthic habitats.

EFH has been designated for multiple life stages of demersal fishes in the Gulf of Maine using local population density derived from long term monitoring surveys as an indirect measure of habitat quality (Reid et al., 1999, Pereira et al., 2012). See MacCall (1990) and Kramer et al. (1997) for discussion and review of the theoretical and empirical basis for this assumption. Data from all years of the surveys were averaged within cells of a 10-min latitude by 10-min longitude square grid (grid cells cover approximately 133 km$^2$ of ocean surface area) and mapped over the continental shelf (Figure 1). EFH alternatives included 100% of the distribution of each species defined by these 10-min square planning units and
subsets of planning units (taking those with the highest densities first) equal to 50%, 75%, and 90% of abundance summed over all planning units. For most species, one of the latter two alternatives was chosen for final EFH designation. Separate EFH designations were developed for the four life-history stages of each species (eggs, larvae, juvenile, and adult). Since most economically-important species are widespread in the Gulf of Maine, EFH for even a single life stage of all species combined covers nearly the entire seafloor (Figure 2). This has posed a significant challenge for the practical implementation of EFH provisions, particularly in prioritizing areas for habitat protection.

Figure 1. Gulf of Maine study area.
In this chapter, we demonstrate how simulated annealing could be used with the survey data set described above to design a small number of protected areas that would meet specific goals for EFH, thereby allowing spatially constrained management actions to reduce the effects of fishing without affecting all fishing activity across the region. We chose the juvenile life stage for analysis because the distributions of juvenile demersal fishes are more tightly linked to seafloor habitat attributes than are other life stages and may therefore benefit the most from habitat protection efforts (Walters and Juanes 1993, Auster and Shackell 2000). We also determine how well chosen planning units would simultaneously protect a substantial portion of overall demersal fish species diversity by quantifying how well they represent species richness (comprehensiveness) and proportion of regional abundance (adequacy) compared to planning units selected at random.

**GEOGRAPHIC CONTEXT AND FISH FAUNA**

Located on the north-eastern continental shelf of the United States, the Gulf of Maine has been described as a unique ecosystem, distinguished oceanographically from other regions of the shelf by differences in bathymetry, mean annual temperature, circulation, seasonal stratification, and productivity (Ingham et al., 1982, Sherman et al., 1996). The Gulf of Maine
lies at the southern extent of the cold temperate, or boreal, province which extends as far north as Newfoundland. A strong zoogeographic boundary at Cape Cod separates many vertebrate and invertebrate Gulf region populations with those of the mid-Atlantic shelf (Cook and Auster 2007). The Northeast Channel with depths exceeding 200m, forms a significant barrier to dispersal of benthic organisms between the Gulf of Maine and the shallower waters of the Scotia Shelf. The demersal fish fauna is composed primarily of year-round residents, with a relatively small number of species migrating seasonally into the area (Auster 2002).

Economically important demersal fishes include species such as Atlantic cod (*Gadus morhua*), haddock (*Melanogrammus aeglefinus*), and yellowtail flounder (*Pleuronectes ferrugineus*) which historically sustained some of the largest fisheries in the world (Kurlanski 1998) (Table 1a). Species not of economic importance, and not targeted in fishery operations, include species such as Gulfstream flounder (*Citharichthys arctifrons*), spotted hake (*Urophycis regia*), and barndoor skate (*Raja laevis*) (Table 1b). Combined, they occupy a wide range of habitats from rocky reef to gravel, sand, or mud bottom over a range of depth and temperature, and both distribution and habitat use can vary with life stage (Auster et al., 2001, Auster et al., 2003a, 2003b).

Wide variation exists in the distribution and abundance of juveniles of economically important species and of non-economically important demersal fishes. The former group has on average, larger geographic distributions in the Gulf of Maine, occupying a mean of 65% of planning units versus 27% by non-economically important species. Juveniles of nine economically important species (60% of that group) occur in more than 60% of planning units compared to 17% of non-economically important species. We define rarity as occupying less than 5% of the region. By this criterion, 18% of economically important species and 28% of non-economically important species are rare in the Gulf of Maine.

**SURVEY DATA**

Data on the distribution and abundance of demersal fishes have been collected in shelf-wide fishery independent bottom trawl surveys in the United States by the National Oceanic and Atmospheric Administration Fisheries Service (NOAA Fisheries) since 1963, and in Canada by the Department of fisheries and Oceans (DFO) since 1970. Sampling by DFO is mainly in the...
eastern Gulf of Maine where there occurs some spatial overlap with NOAA Fisheries surveys. Both utilize a stratified random sampling design with common sets of strata defined by depth (varying from 30 to 400 m) and latitude (Clark and Brown 1977).

Table 1. Demersal fishes in the Gulf of Maine. Percent total area is the percent of 10-min lat/long square grid cells in which the species has been detected in long term monitoring surveys. Regional abundance is the sum of transformed mean abundances calculated per grid cell over all surveys [See Cook and Auster (2012) for details]. R: Resident. SM: Southern Migrant. NM: Northern Migrant

<table>
<thead>
<tr>
<th>Common Name</th>
<th>Scientific Name</th>
<th>Status</th>
<th>Percent total area</th>
<th>Regional abundance</th>
</tr>
</thead>
<tbody>
<tr>
<td>A) Economically important species</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Acadian redfish</td>
<td><em>Sebastes faciatus</em></td>
<td>R</td>
<td>95.43</td>
<td>307.67</td>
</tr>
<tr>
<td>American plaice</td>
<td><em>Hippoglossus platessoides</em></td>
<td>R</td>
<td>96.57</td>
<td>327.86</td>
</tr>
<tr>
<td>Atlantic cod</td>
<td><em>Gadus morhua</em></td>
<td>R</td>
<td>57.43</td>
<td>76.71</td>
</tr>
<tr>
<td>Goosefish</td>
<td><em>Lophius americanus</em></td>
<td>R</td>
<td>80.86</td>
<td>46.22</td>
</tr>
<tr>
<td>Haddock</td>
<td><em>Melanogrammus aeglefinus</em></td>
<td>R</td>
<td>82.57</td>
<td>141.08</td>
</tr>
<tr>
<td>Ocean pout</td>
<td><em>Macrozoarces americanus</em></td>
<td>R</td>
<td>54.86</td>
<td>22.48</td>
</tr>
<tr>
<td>Pollock</td>
<td><em>Pollachius virens</em></td>
<td>R</td>
<td>78.00</td>
<td>88.93</td>
</tr>
<tr>
<td>Red hake</td>
<td><em>Urophycis chuss</em></td>
<td>R</td>
<td>96.57</td>
<td>154.41</td>
</tr>
<tr>
<td>Silver hake</td>
<td><em>Merluccius bilinearis</em></td>
<td>R</td>
<td>99.71</td>
<td>648.88</td>
</tr>
<tr>
<td>Spiny dogfish</td>
<td><em>Squalus acanthias</em></td>
<td>R</td>
<td>6.57</td>
<td>1.70</td>
</tr>
<tr>
<td>White hake</td>
<td><em>Urophycis tenuis</em></td>
<td>R</td>
<td>96.00</td>
<td>158.54</td>
</tr>
<tr>
<td>Windowpane flounder</td>
<td><em>Scopthalmus aquosus</em></td>
<td>R</td>
<td>12.29</td>
<td>4.88</td>
</tr>
<tr>
<td>Winter flounder</td>
<td><em>Pleuronectes americanus</em></td>
<td>R</td>
<td>19.14</td>
<td>18.03</td>
</tr>
<tr>
<td>Witch flounder</td>
<td><em>Glyptocephalus cynoglossus</em></td>
<td>R</td>
<td>78.86</td>
<td>93.16</td>
</tr>
<tr>
<td>Yellowtail flounder</td>
<td><em>Pleuronectes ferrugineus</em></td>
<td>R</td>
<td>25.14</td>
<td>21.91</td>
</tr>
<tr>
<td>B) Non-economically important species</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alligatorfish</td>
<td><em>Aspidophoroides monopterygius</em></td>
<td>R</td>
<td>32.86</td>
<td>41.24</td>
</tr>
<tr>
<td>Atlantic hagfish</td>
<td><em>Myxine glutinosa</em></td>
<td>R</td>
<td>58.57</td>
<td>76.36</td>
</tr>
<tr>
<td>Atlantic wolffish</td>
<td><em>Anarrhichas lupis</em></td>
<td>R</td>
<td>59.43</td>
<td>92.38</td>
</tr>
<tr>
<td>Barndoor skate</td>
<td><em>Raja laevis</em></td>
<td>R</td>
<td>7.43</td>
<td>3.51</td>
</tr>
<tr>
<td>Common Name</td>
<td>Scientific Name</td>
<td>Status</td>
<td>Percent total area</td>
<td>Regional abundance</td>
</tr>
<tr>
<td>----------------------</td>
<td>----------------------------</td>
<td>--------</td>
<td>--------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>Clearnose skate</td>
<td>Raja eglanteria</td>
<td>R</td>
<td>12.86</td>
<td>8.03</td>
</tr>
<tr>
<td>Cunner</td>
<td>Tautogobius adspersus</td>
<td>R</td>
<td>9.14</td>
<td>8.74</td>
</tr>
<tr>
<td>Cusk</td>
<td>Brosme brosme</td>
<td>R</td>
<td>79.43</td>
<td>233.85</td>
</tr>
<tr>
<td>Daubed shanney</td>
<td>Lumpenus maculatus</td>
<td>R</td>
<td>26.86</td>
<td>185.73</td>
</tr>
<tr>
<td>Eelpout</td>
<td>Lycodonys mirabilis</td>
<td>R</td>
<td>1.43</td>
<td>0.74</td>
</tr>
<tr>
<td>Fawn cusk eel</td>
<td>Lepophidium cervinum</td>
<td>R</td>
<td>5.43</td>
<td>3.66</td>
</tr>
<tr>
<td>Four beard rockling</td>
<td>Enchelopus cimbrius</td>
<td>R</td>
<td>62.86</td>
<td>151.14</td>
</tr>
<tr>
<td>Fourspot flounder</td>
<td>Paralichthys oblongus</td>
<td>R</td>
<td>22.57</td>
<td>38.85</td>
</tr>
<tr>
<td>Grubby sculpin</td>
<td>Myoxocephalus anaeus</td>
<td>R</td>
<td>2.57</td>
<td>1.65</td>
</tr>
<tr>
<td>Gulfstream flounder</td>
<td>Citharichthys arcticfons</td>
<td>R</td>
<td>4.86</td>
<td>2.40</td>
</tr>
<tr>
<td>Little skate</td>
<td>Raja erinacea</td>
<td>R</td>
<td>36.29</td>
<td>144.62</td>
</tr>
<tr>
<td>Longhorn sculpin</td>
<td>Myoxocephalus octodecemspinia</td>
<td>R</td>
<td>63.71</td>
<td>1070.80</td>
</tr>
<tr>
<td>Lumpfish</td>
<td>Cyclopterus lumpus</td>
<td>R</td>
<td>30.00</td>
<td>11.17</td>
</tr>
<tr>
<td>Mustache sculpin</td>
<td>Triglops murrayi</td>
<td>R</td>
<td>33.71</td>
<td>103.10</td>
</tr>
<tr>
<td>Northern sand lance</td>
<td>Ammodytes dubius</td>
<td>R</td>
<td>14.00</td>
<td>3496.45</td>
</tr>
<tr>
<td>Northern searobin</td>
<td>Prionotus carolinus</td>
<td>R</td>
<td>4.00</td>
<td>1.54</td>
</tr>
<tr>
<td>Radiated shanney</td>
<td>Ulvaria subbifurcata</td>
<td>R</td>
<td>14.86</td>
<td>84.28</td>
</tr>
<tr>
<td>Rock gunnel</td>
<td>Pholis gunnellus</td>
<td>R</td>
<td>2.86</td>
<td>3.07</td>
</tr>
<tr>
<td>Sea lamprey</td>
<td>Petromyzon marinus</td>
<td>R</td>
<td>4.00</td>
<td>1.10</td>
</tr>
<tr>
<td>Sea raven</td>
<td>Hemitripterus americanus</td>
<td>R</td>
<td>74.00</td>
<td>245.15</td>
</tr>
<tr>
<td>Seasnail</td>
<td>Liparis atlanticus</td>
<td>R</td>
<td>5.14</td>
<td>3.14</td>
</tr>
<tr>
<td>Shorthorn sculpin</td>
<td>Myoxocephalus scorpius</td>
<td>R</td>
<td>1.43</td>
<td>0.87</td>
</tr>
<tr>
<td>Smooth dogfish</td>
<td>Mustelus canis</td>
<td>SM</td>
<td>0.86</td>
<td>0.53</td>
</tr>
<tr>
<td>Smooth skate</td>
<td>Raja senta</td>
<td>R</td>
<td>90.00</td>
<td>230.48</td>
</tr>
<tr>
<td>Snake blenny</td>
<td>Lumpenus lampretaeformis</td>
<td>R</td>
<td>16.86</td>
<td>29.59</td>
</tr>
<tr>
<td>Snowflake hookear sculpin</td>
<td>Artediiellus incinatus</td>
<td>R</td>
<td>6.57</td>
<td>9.33</td>
</tr>
<tr>
<td>Spotted hake</td>
<td>Urophycis regia</td>
<td>SM</td>
<td>4.00</td>
<td>2.07</td>
</tr>
<tr>
<td>Thorny skate</td>
<td>Raja radiata</td>
<td>R</td>
<td>99.71</td>
<td>856.53</td>
</tr>
<tr>
<td>Threespine stickleback</td>
<td>Gasterosteus aculeatus</td>
<td>R</td>
<td>14.29</td>
<td>8.17</td>
</tr>
<tr>
<td>Vahl's checker eelpout</td>
<td>Lycodes vahlil</td>
<td>NM</td>
<td>2.57</td>
<td>0.78</td>
</tr>
<tr>
<td>Winter skate</td>
<td>Raja ocellata</td>
<td>R</td>
<td>36.57</td>
<td>133.67</td>
</tr>
<tr>
<td>Wrymouth</td>
<td>Cryptacanthodes maculatus</td>
<td>R</td>
<td>19.71</td>
<td>11.22</td>
</tr>
</tbody>
</table>
Stations within strata (4730 total in the Gulf of Maine) were assigned at random for each survey. Surveys were conducted in all seasons but mostly in summer and autumn. The number of stations allotted to a stratum is in proportion to its geographic area with approximately one station per 200 nm$^2$. Additional sampling is also conducted in nearshore waters as shallow as 9 m. At each station, a single tow of $\frac{1}{2}$-h duration is made along the bottom at a speed of 10.5 km hr$^{-1}$ (3.5 nm hr$^{-1}$). All fish captured are identified, counted, and measured to the nearest cm.

We followed the methods used by Reid et al. (1999) (in the mapping of EFH described above) to prepare for analysis, the raw survey data for juveniles of economically important species collected from 1963-2000. We omitted Atlantic halibut (*Hippoglossus hippoglossus*) from the analysis because they occur currently in such small numbers that historical distribution data are likely to be an unreliable indication of current status. We used length at maturity estimates (i.e., the length at which 50% of individuals are reproductively mature) from Reid et al. (1999) to separate juvenile and adult fish data records.

Data for the remainder of the demersal fish fauna were compiled from a combination of NOAA Fisheries and DFO trawl surveys for the years 1970-1994 by Brown et al. (1996) for the East Coast of North America Strategic Assessment Project (ECNASAP). We grouped fish species into demersal, pelagic, and mesopelagic classes and by residency status. Of these, we chose for analysis, the 36 demersal species that were resident or seasonal migrants to the Gulf of Maine (Table 1b). The ECNASAP data set does not track individuals by length, so counts consisted of all age classes. See Cook and Auster (2012) for further details on the treatment of both data sets.

**EFH DESIGN SCENARIOS**

We set conservation goals, or targets, for each species as a percentage of total regional abundance which is the summed mean abundance over all planning units in which the species occurs. Using a conservation goal of 20% of regional abundance per species, we explored a variety of spatial design scenarios including those with no spatial constraints (BLM = 0), and those with BLM values high enough to obtain solutions with a single aggregate of planning units (i.e., a single potential protected area) and solutions with a moderate number (2-4) of aggregated sites.
We set the cost of all planning units to be equal although they could potentially be weighted by loss of fishing opportunity, restrictions on certain gear types, or other socio-economic factors. We set the penalties high to ensure conservation targets were met for all species (i.e., sum of the penalties was zero) and assigned no threshold penalty, such that the total cost represented a balance between total area and boundary length. The lowest cost solutions were therefore composed of the smallest number of planning units in which all conservation goals were met under constraints set by the BLM. We used an adaptive annealing schedule with 10,000 steps and one million iterations per run, and finished each run with normal iterative improvement (a Marxan procedure that removes nonessential planning units and helps ensure solutions are close to a global optimum (Ball and Possingham 2001). We ran the algorithm 100 times under each scenario, saving the lowest cost (i.e., best) solutions in each case.

RESULTS

Spatially unconstrained solutions (BLM = 0) contain numerous small clusters and isolated planning units. By experimentally adjusting the BLM, we were able to find solutions with planning units aggregated into a single site as well as solutions of three and four sites. The best solutions of all aggregate levels covered less than 20% of the Gulf of Maine (Figure 3). The amount of total habitat needed to meet the same conservation goal was greater with higher degrees of spatial aggregation (i.e., single-site solutions were larger that four-site solutions).

Simulated annealing was highly efficient in including planning units with the highest densities of juveniles of most economically important species. This can be demonstrated by comparing the percent of planning units occupied by species at their highest densities with the percent of all occupied planning units included in solutions. For the best four site solution, the percent of the highest density planning units, those that rank in the upper 10th percentile of cumulative regional abundance, exceeded the percentage of all habitat represented for 13 of the 15 species analyzed (Figure 4). A full fifty percent or more of the highest density habitat for eight of the 15 species was included in this solution.
To determine how well the best solutions performed with respect to capturing overall demersal fish species diversity, we compared species richness (or comprehensiveness) and percent of regional abundance (or adequacy) against the mean values of these quantities derived from 100 sets of randomly drawn planning units equal in number to those in the solutions. Large percentages (86-94%) of the demersal fish fauna were represented in all solutions (Table 2), and the mean number of species represented in planning units drawn at random was significantly greater only in the non-constrained solution. Further, non-economically important species occurred at greater abundances in all solutions compared with randomly drawn planning units, and these solutions contained a significantly larger number of non-economically important species at abundance levels exceeding the conservation goals set for juveniles of economically important species than randomly drawn squares (Table 3).

Six non-economically important species were absent from the best solutions of one or more scenarios [Shorthorn sculpin (*Myoxocephalus scorpius*), smooth dogfish (*Mustelus canis*), Vahl’s checker eelpout (*Lycodes vahlii*), snowflake hookear sculpin (*Arctediellus uncinatus*), spotted hake (*Urophycis regia*), northern searobin (*Prionotus carolinus*]. All are at the northern or southern edge of their geographic distributions and rare within the Gulf of Maine by our definition, except the snowflake hookear sculpin,
reported from 7.43% of planning units. The rarest species, smooth dogfish, was reported from only 0.86% of squares.

Figure 4. The best solutions obtained in each of three sets of 100 runs of Marxan for a 20% conservation goal. A) no spatial constraints (BLM = 0); B) the best four-site solution identified by increasing the BLM; C) the best single-site solution identified by increasing the BLM even more. The outer boundary represents the extent of the trawl survey at approximately 100m bathymetric contour.
Table 2. Comparisons between numbers of non-economically important demersal fish species represented by each of three simulated annealing scenarios (Observed) versus mean numbers of species in equivalent numbers of planning units drawn at random (Expected)

<table>
<thead>
<tr>
<th>Number of Sites</th>
<th>Percent total area</th>
<th>Observed (total/ percent)</th>
<th>Expected (total/ percent)</th>
<th>sd expected</th>
<th>z score</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>14.00</td>
<td>31/86</td>
<td>34/94</td>
<td>1.598</td>
<td>-2.183</td>
<td>&lt;0.05</td>
</tr>
<tr>
<td>4</td>
<td>15.71</td>
<td>34/94</td>
<td>35/97</td>
<td>1.356</td>
<td>-0.590</td>
<td>&gt;0.05</td>
</tr>
<tr>
<td>1</td>
<td>19.71</td>
<td>34/94</td>
<td>36/100</td>
<td>1.029</td>
<td>-1.507</td>
<td>&gt;0.05</td>
</tr>
</tbody>
</table>

Table 3. Numbers of non-economically important demersal fish species present at equal to or greater than (A) 10%, (B) 20%, and (C) 30% of total regional abundance within the best solutions of three simulation scenarios (Observed) versus randomly drawn (Expected) planning units

<table>
<thead>
<tr>
<th>Number of Sites</th>
<th>(A)</th>
<th>(B)</th>
<th>(B)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Observed</td>
<td>Expected</td>
<td>z score</td>
</tr>
<tr>
<td>32</td>
<td>5</td>
<td>2</td>
<td>2.204</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>2</td>
<td>1.086</td>
</tr>
<tr>
<td>1</td>
<td>15</td>
<td>4</td>
<td>5.087</td>
</tr>
</tbody>
</table>

**DISCUSSION**

The mandate for fishery managers under the Sustainable Fisheries Act of 1996 is to minimize, to the extent practicable, the effects of fishing on EFH. However, achieving such a goal is especially difficult when EFH covers most of the continental shelf. Here we have demonstrated how simulated annealing could be used to identify the minimum amount of habitat needed to satisfy a specific set of conservation goals. Since we defined these goals as a proportion of cumulative abundance, efficiency in site selection, with respect to reducing the total area included in solutions, was achieved by preferentially selecting planning units with high densities of one or more species. We conclude that simulated annealing is a viable and flexible decision support tool for informing the design of EFH in such a way that would satisfy specific objectives for
Simulated Annealing and Fish Habitat Conservation while allowing spatially constrained management actions to reduce the effects of fishing without affecting all fishing activity across a region. In the Gulf of Maine, MPAs designed with the aid of simulated annealing to enhance fisheries could also benefit a large number of non-economically important species, which suggests that the same habitat attributes that confer greater survivorship among juveniles of some species also benefit many other species in the community.

Our results for the Gulf of Maine illustrate how simulated annealing can be used to guide spatial planning for the enhancement of fisheries as well as the conservation of non-economically important species that share similar habitats. They contrast with those of other studies such as Gladstone’s (2007) because, we believe, the distributions of juveniles of economically important species are relatively large and tend to overlap those of other demersal fishes and because spatial turnover in species composition in the cool temperate regions of the higher latitudes is lower than that of the more diverse marine ecosystems of the tropics and warm temperate regions of the world. Such general conclusions cannot be drawn from a few case studies, but deserve further study.

Here, we assumed that the goal was to optimize representation of habitat with the greatest flexibility afforded to site selection. As a result, some solutions contained few or no planning units of high density of some species. Greater representation of high density habitat can be achieved by increasing conservation targets and/or relaxing spatial constraints by reducing the BLM. Another approach would limit analysis to planning units containing only higher densities of one or more species. Alternatively, one could perform a two-step optimization procedure that would first identify habitat for a subset of species (e.g., those most sensitive to disturbance), then using these sites as a starting point, complete representation for all other species with the inclusion of additional habitat.

An important aspect of simulated annealing is its ability to identify multiple solutions of similar cost. Although we presented results for the best solution as defined by the objective function above, other solutions that were nearly as good were also found. Multiple solutions give decision-makes a great deal of flexibility in the planning process, enabling them to balance other priorities with habitat conservation. Marxan further produces a summed solution that includes a count of the number of times each planning unit is part of one or more solutions, enabling users to identify which planning units are necessary for meeting stated conservation goals.
Figure 5. Proportions of “high density habitat” (defined as the upper 10th, 20th, and 30th percentiles of regional abundance versus all habitat represented for juveniles of economically-important demersal fish species in the best of 100 Marxan runs with a 20% (summed regional abundance) conservation goal and moderate degree of aggregation (four sites) in the Gulf of Maine.

Another important aspect of simulated annealing is its ability to incorporate explicit socioeconomic variables into the cost of planning units (Sala et al., 2002, Ban et al., 2009). These approaches are becoming more popular, and are especially important in marine spatial planning because ecological criteria and socioeconomic measures are not independent where impacts from fishery operations are a significant threat (Castilla 1999).

Ongoing management problems for Gulf of Maine fisheries (e.g., Bryce 2013) could benefit from the use of simulated annealing to develop spatial management options that simultaneously address multiple management goals (NEFMC 2013). For example, Gulf of Maine Atlantic cod populations, the mainstay of the fishery, are heavily over-fished. As a result, an extremely low quota has been set which in turn minimizes fishing opportunities for other species if cod are caught as bycatch. A change to the quota management approach has given some the perception that closed areas for habitat conservation are now unnecessary as minimal bottom contact time for fishing gear would conserve seafloor habitat. However, this is counter to the assumption that fish are distributed based on density dependent habitat selection theory (Auster et al. 2001, Pereira et al. 2013), where such an approach works against long term sustainability of these exploited populations by impacting the most important habitat. Development of a revised omnibus habitat amendment (now underway and one that applies to multiple fisheries...
management plans) utilizes hotspots for the young-of-year life history stage of all managed species (versus the entire juvenile period as we used above), with over-fished species given greater weighting factors, while linking those distributions to the most vulnerable habitats (NEFMC 2011, 2013). An iterative process using threshold values based on statistical distributions was used to identify options for closed area alternatives. While such an approach clearly produced useful alternatives for evaluation, simulated annealing approaches could have been implemented to initially evaluate trade-offs in site selection based on density-dependent distributions, vulnerable habitats, and economic impacts (as e.g., measured by distribution of fishing effort).

The later factor is of particular interest for example, to the inshore small boat fleet of fishermen as a number of options have created obstacles to small inshore vessels such that they would need to work further offshore to find fishing opportunities. Economic consequences of habitat alternatives are normally only considered during a post-hoc analysis after the alternatives have been set rather than during the alternative development process.

Finally, a subsequent process to evaluate the distribution of deep sea corals and develop protective measures (NOAA 2010) is currently scheduled to commence after the habitat amendment is completed. Simulated annealing would have been a useful tool to integrate all of these management needs into a single set of analyses to consider trade-offs and offer a range of optimal solutions.

Although fishery management and broader ocean conservation issues are often fraught with conflict due to the disparate needs of a diversity of stakeholders (Hilborn 2007), transparent analytical approaches like simulated annealing that are clearly viewed to optimize the goals of all stakeholders have the potential to minimize conflict and create an environment of reasoned debate on the trade-offs necessary to implement management regimes for the conservation and sustainable use of marine biological diversity.

ACKNOWLEDGMENTS

We thank the National Oceanic and Atmospheric Administration, Northeast Fisheries Science Center at Woods Hole for providing trawl survey data; F. Almeida, J. Palmer, and C. Zetlin for assistance with the data; I. Ball and H. Possingham for permission to use Marxan; K. Joy for technical assistance. Funding was provided by the National Oceanic and Atmospheric Administration's Stellwagen Bank National Marine Sanctuary, the National
Marine Sanctuary Foundation, the Conservation Law Foundation, Connecticut Sea Grant, and Oceana. The opinions expressed herein are those of the authors and do not necessarily reflect the opinions of any of the funding agencies or institutions.

REFERENCES


Complimentary Contributor Copy


Complimentary Contributor Copy


### INDEX

| A | benefits, 216 |
|   | bias, 40 |
|   | biodiversity, 222, 223, 238, 239 |
|   | biomass, 239 |
|   | blogs, 239 |
|   | Boltzmann constant, 9, 173 |
|   | Boltzmann distribution, 102, 143 |
|   | bounds, 35, 38, 40, 41, 43, 106, 140, 141, 146, 151 |
|   | Brazil, ix, 55, 56, 95 |
|   | breeding, 224 |
|   | Broadband, 170, 176 |
|   | access, 196, 206 |
|   | accessibility, 192 |
|   | accounting, 192 |
|   | acetic acid, 157 |
|   | acid, 72, 157 |
|   | adaptation, 62, 194, 198, 216 |
|   | adjustment, 10, 210, 214, 215 |
|   | adverse effects, 224 |
|   | aerospace, 56 |
|   | aerospace engineering, 56 |
|   | agencies, 238 |
|   | aggregation, 224, 231, 236 |
|   | amino, 94, 149, 157 |
|   | amino acid, 94 |
|   | amplitude, 63 |
|   | ancestors, 72 |
|   | ANOVA, 85, 210 |
|   | aqueous solutions, 149 |
|   | Argentina, 35 |
|   | Asia, 93 |
|   | AT&T, 32 |
|   | Austria, 217 |
| B | base, 100, 103, 107, 194, 223, 238 |
|   | Beijing, 135 |
|   | benchmarks, 82 |
|   | C++ , 81, 200, 209 |
|   | CAD, ix |
|   | Cairo, 139 |
|   | calibration, 94 |
|   | CAM, ix |
|   | candidates, 62 |
|   | carbon, 157 |
|   | carbon dioxide (CO2), 145, 149, 154, 156, 157 |
|   | case study(ies), 70, 95, 209, 217, 219, 235 |
|   | challenges, 56, 241 |
|   | chaos, 113 |
|   | chemical, viii, 117, 139, 140, 144, 147, 148, 150, 154, 155 |

Complimentary Contributor Copy
chemical equilibrium, viii, 139, 140, 147, 154
chemical reactions, 150
Chicago, 96
children, 25
Chile, 194
China, 99
circulation, 226
classes, 119, 230
classification, 217, 239
clusters, 231
coding, 31, 94
color, 42, 52, 72, 95
commercial, 192, 194
community(ies), 36, 222, 223, 235, 238, 239
comparative analysis, 19
competition, 55, 56
competitiveness, 55
complexity, 51, 73, 74, 77, 147, 214, 216
composition, 145, 146, 147, 150, 235, 238
computation, 24, 73, 88, 104, 133, 151, 157
computer, ix, 8, 11, 14, 28, 101, 111, 121, 153, 209
computing, 24, 29, 32, 142, 155, 190, 203, 205
configuration, 2, 41, 55, 58, 67, 77, 82, 88, 102, 131
conflict, 237
Congress, 94, 137, 153, 157
conjugate gradient method, 28
connectivity, 222
crystallization, vii, 56, 57, 63, 64, 65
crystals, 1
crystal structure, 2, 28
crystals, vii, 63, 65, 66, 72
cost, 2, 5, 11, 12, 43, 51, 62, 63, 65, 66, 72
consuming, viii, 119, 121, 132
daily transport tasks, viii, 191, 195, 209
data distribution, 85
data set, 209, 210, 226, 230
database, 18, 19, 194
decay, 173
decomposition, vii, 36, 39
decoupling, 162, 163, 179, 189
defects, 1
denmark, 32
department of agriculture, 217
department of commerce, 240, 241
dependent variable, 148
depth, 227, 228
<table>
<thead>
<tr>
<th>Term</th>
<th>Page References</th>
</tr>
</thead>
<tbody>
<tr>
<td>derivatives</td>
<td>102</td>
</tr>
<tr>
<td>designers</td>
<td>127</td>
</tr>
<tr>
<td>deviation</td>
<td>12, 41, 87, 111, 122, 210, 211</td>
</tr>
<tr>
<td>difference threshold</td>
<td>20</td>
</tr>
<tr>
<td>differential equations</td>
<td>145</td>
</tr>
<tr>
<td>dimensionality</td>
<td>147, 150</td>
</tr>
<tr>
<td>dispersion</td>
<td>43</td>
</tr>
<tr>
<td>displacement</td>
<td>118</td>
</tr>
<tr>
<td>distillation</td>
<td>148, 156</td>
</tr>
<tr>
<td>distribution</td>
<td>7, 9, 10, 20, 32, 42, 43, 48, 63, 102, 143, 147, 223, 225, 227, 230, 237, 240</td>
</tr>
<tr>
<td>distribution function</td>
<td>20</td>
</tr>
<tr>
<td>diversification</td>
<td>142, 151</td>
</tr>
<tr>
<td>diversity</td>
<td>105, 107, 222, 223, 226, 232, 237, 238, 241</td>
</tr>
<tr>
<td>DNA</td>
<td>vii, 71, 72, 94</td>
</tr>
<tr>
<td>ecosystem</td>
<td>226</td>
</tr>
<tr>
<td>Egypt</td>
<td>139</td>
</tr>
<tr>
<td>electrolyte</td>
<td>149</td>
</tr>
<tr>
<td>energy</td>
<td>1, 2, 9, 28, 102, 140, 142, 143, 145, 146, 147, 148, 150, 153, 154, 155, 162, 173</td>
</tr>
<tr>
<td>enforcement</td>
<td>223</td>
</tr>
<tr>
<td>engineering</td>
<td>ix, 56, 109, 141, 144, 148, 155, 172</td>
</tr>
<tr>
<td>England</td>
<td>239, 240</td>
</tr>
<tr>
<td>entrapment</td>
<td>140</td>
</tr>
<tr>
<td>entropy</td>
<td>38, 46, 49, 52, 143</td>
</tr>
<tr>
<td>environment</td>
<td>28, 121, 237</td>
</tr>
<tr>
<td>equality</td>
<td>35, 36, 38, 39, 44, 45, 48, 49, 51, 53, 119, 146, 150</td>
</tr>
<tr>
<td>equilibrium</td>
<td>vii, 21, 102, 139, 140, 142, 145, 146, 147, 148, 149, 150, 152, 154, 155, 156, 157, 204</td>
</tr>
<tr>
<td>equipment</td>
<td>148</td>
</tr>
<tr>
<td>estimation problems</td>
<td>149</td>
</tr>
<tr>
<td>ethanol</td>
<td>157</td>
</tr>
<tr>
<td>evolution</td>
<td>37, 49, 52, 62, 65, 66, 69, 94, 95, 101, 110, 143, 147, 155, 156, 173</td>
</tr>
<tr>
<td>evolutionary computation</td>
<td>24, 36</td>
</tr>
<tr>
<td>exclusion</td>
<td>239</td>
</tr>
<tr>
<td>execution</td>
<td>63, 80, 100, 110, 121</td>
</tr>
<tr>
<td>exploitation</td>
<td>106, 151</td>
</tr>
<tr>
<td>fauna</td>
<td>227, 230, 232</td>
</tr>
<tr>
<td>feature selection</td>
<td>144, 153</td>
</tr>
<tr>
<td>financial support</td>
<td>189</td>
</tr>
<tr>
<td>Finland</td>
<td>194, 220</td>
</tr>
<tr>
<td>fish</td>
<td>vii, 221, 222, 223, 224, 226, 227, 230, 232, 234, 236, 238, 241</td>
</tr>
<tr>
<td>fisheries</td>
<td>vii, 221, 222, 223, 224, 227, 235, 236, 238, 239</td>
</tr>
<tr>
<td>fishing</td>
<td>222, 224, 226, 231, 234, 236, 237</td>
</tr>
<tr>
<td>fitness</td>
<td>9, 11, 25, 76</td>
</tr>
<tr>
<td>flexibility</td>
<td>61, 235</td>
</tr>
<tr>
<td>flight</td>
<td>67, 101</td>
</tr>
<tr>
<td>fluctuations</td>
<td>86</td>
</tr>
<tr>
<td>fluid</td>
<td>viii, 157, 159, 162, 171, 179</td>
</tr>
<tr>
<td>force</td>
<td>28, 35</td>
</tr>
<tr>
<td>formation</td>
<td>35, 142</td>
</tr>
<tr>
<td>formula</td>
<td>6, 7, 15, 20, 80, 114</td>
</tr>
<tr>
<td>France</td>
<td>71</td>
</tr>
<tr>
<td>free energy</td>
<td>1, 102, 140, 145, 146, 147, 153, 155</td>
</tr>
<tr>
<td>fuel consumption</td>
<td>viii, 119, 121</td>
</tr>
<tr>
<td>function evaluation</td>
<td>vii, 46, 47, 49, 51, 109, 114, 116</td>
</tr>
<tr>
<td>function values</td>
<td>2, 3, 5, 6, 7, 11, 14, 15, 20, 25, 26, 142</td>
</tr>
<tr>
<td>funding</td>
<td>238</td>
</tr>
<tr>
<td>genetics</td>
<td>72</td>
</tr>
<tr>
<td>genome</td>
<td>72</td>
</tr>
<tr>
<td>genotype</td>
<td>97</td>
</tr>
<tr>
<td>geometry</td>
<td>56, 58, 60, 67, 117</td>
</tr>
<tr>
<td>Gibbs energy</td>
<td>145, 147, 148, 155</td>
</tr>
<tr>
<td>GIS</td>
<td>194</td>
</tr>
<tr>
<td>gravity</td>
<td>58</td>
</tr>
<tr>
<td>greenhouse</td>
<td>219</td>
</tr>
<tr>
<td>greenhouse gas emissions</td>
<td>219</td>
</tr>
<tr>
<td>growth</td>
<td>224</td>
</tr>
<tr>
<td>H</td>
<td>L</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>habitat(s), 222, 224, 225, 226, 227, 231, 234, 235, 236, 237, 238, 240, 241</td>
<td>Lagrange multipliers, 37</td>
</tr>
<tr>
<td>habitat quality, 224</td>
<td>landscape, 102</td>
</tr>
<tr>
<td>harvesting, 192</td>
<td>larvae, 225</td>
</tr>
<tr>
<td>height, 58, 172</td>
<td>laws, 143</td>
</tr>
<tr>
<td>homologous genes, 72</td>
<td>lead, 51, 72, 88, 90, 141</td>
</tr>
<tr>
<td>hybrid, 2, 3, 18, 20, 21, 22, 24, 27, 29, 31, 96, 100, 117, 139, 141, 144, 153, 190</td>
<td>legs, 72</td>
</tr>
<tr>
<td>hybridization, 144</td>
<td>light, 56</td>
</tr>
<tr>
<td>hypercube, 8</td>
<td>linear programming, 39</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>I</th>
<th>M</th>
</tr>
</thead>
<tbody>
<tr>
<td>image, ix, 32, 100</td>
<td>macromolecules, 72</td>
</tr>
<tr>
<td>improvements, 15, 127, 143, 204, 205</td>
<td>magnitude, 118, 126, 127, 128, 130, 131, 133</td>
</tr>
<tr>
<td>impulses, 124, 125, 126, 131, 133</td>
<td>management, 192, 217, 218, 220, 222, 226, 235, 236, 237, 239</td>
</tr>
<tr>
<td>impulsive, 99, 101, 124, 133, 138</td>
<td>manifolds, 151</td>
</tr>
<tr>
<td>independent variable, 125, 148, 151</td>
<td>mapping, 59, 230</td>
</tr>
<tr>
<td>India, 157</td>
<td>marine fish, 222, 224</td>
</tr>
<tr>
<td>indirect measure, 224</td>
<td>Markov chain, 5, 11, 15, 16, 80, 143</td>
</tr>
<tr>
<td>industry(ies), 192, 215, 217, 219, 220</td>
<td>Maryland, 239, 241</td>
</tr>
<tr>
<td>inequality, 35, 36, 38, 39, 42, 44, 45, 47, 48, 49, 51</td>
<td>mass, 8, 118, 119, 120, 122, 146, 147</td>
</tr>
<tr>
<td>initiation, 118</td>
<td>matrix, vii, 36, 39, 40, 42, 44, 45, 78, 159, 161, 162, 163, 165, 166, 167, 168, 179, 182, 183, 185, 186, 187, 188</td>
</tr>
<tr>
<td>insertion, 77, 205, 206, 207, 208, 213, 214, 216</td>
<td>measurement, 148</td>
</tr>
<tr>
<td>institutions, 238</td>
<td>measurements, 60</td>
</tr>
<tr>
<td>integrated circuits, 100</td>
<td>medical, ix</td>
</tr>
<tr>
<td>investment, 192</td>
<td>melts, 102</td>
</tr>
<tr>
<td>iteration, 4, 5, 8, 12, 36, 38, 40, 41, 43, 46, 48, 50, 51, 62, 63, 78, 103, 105, 113, 126, 160, 174, 205, 213</td>
<td>memory, 209</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>J</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Japan, ix</td>
<td>messages, 16</td>
</tr>
<tr>
<td>juveniles, 226, 227, 230, 231, 232, 235, 236</td>
<td>metallurgy, 1</td>
</tr>
<tr>
<td></td>
<td>methanol, 148</td>
</tr>
<tr>
<td></td>
<td>methodology, 55, 75, 82, 92, 158</td>
</tr>
<tr>
<td></td>
<td>Mexico, 97</td>
</tr>
<tr>
<td></td>
<td>migrants, 230</td>
</tr>
</tbody>
</table>

Complimentary Contributor Copy
<table>
<thead>
<tr>
<th>Index</th>
<th>247</th>
</tr>
</thead>
<tbody>
<tr>
<td>minimum fuel consumption, viii, 119, 121</td>
<td>operations, 51, 93, 104, 192, 194, 195, 216, 217, 220, 222, 224, 227, 236</td>
</tr>
<tr>
<td>minimum transfer time, viii, 119, 120</td>
<td>opportunism, 240</td>
</tr>
<tr>
<td>MIP, 194, 215</td>
<td>opportunities, 236, 237, 241</td>
</tr>
<tr>
<td>mission(s), 101, 135</td>
<td>optimal performance, 151</td>
</tr>
<tr>
<td>mixing, 146, 149, 157</td>
<td>optimization method, 2, 3, 6, 8, 18, 24, 28, 124, 135, 138, 141, 150, 152, 155, 156</td>
</tr>
<tr>
<td>modelling, 2</td>
<td>orbit, vii, 99, 117, 118, 119, 120, 128, 132, 136, 137</td>
</tr>
<tr>
<td>models, 41, 46, 47, 56, 57, 59, 60, 63, 70, 94, 126, 139, 140, 148, 154, 194</td>
<td>organize, 56, 192</td>
</tr>
<tr>
<td>modifications, 36</td>
<td>overlap, 131, 228, 235</td>
</tr>
<tr>
<td>mole, 147</td>
<td>ownership, 192</td>
</tr>
<tr>
<td>molecular structure, 33</td>
<td></td>
</tr>
<tr>
<td>momentum, 180, 181</td>
<td></td>
</tr>
<tr>
<td>Monte Carlo method, 2, 143</td>
<td></td>
</tr>
<tr>
<td>MPI, 121, 133</td>
<td></td>
</tr>
<tr>
<td>multiplication, 166, 167, 168</td>
<td></td>
</tr>
<tr>
<td>mutation, 104, 105, 111</td>
<td></td>
</tr>
<tr>
<td>natural selection, 241</td>
<td></td>
</tr>
<tr>
<td>negativity, 200</td>
<td></td>
</tr>
<tr>
<td>neutral, 28</td>
<td></td>
</tr>
<tr>
<td>New England, 239, 240</td>
<td></td>
</tr>
<tr>
<td>New South Wales, 220</td>
<td></td>
</tr>
<tr>
<td>New Zealand, 209</td>
<td></td>
</tr>
<tr>
<td>next generation, 145</td>
<td></td>
</tr>
<tr>
<td>NMR, 28</td>
<td></td>
</tr>
<tr>
<td>NOAA, 227, 230, 237, 240, 241</td>
<td></td>
</tr>
<tr>
<td>nodes, 72, 73, 76, 78, 165, 166, 167, 168, 179, 180, 186, 188, 193</td>
<td></td>
</tr>
<tr>
<td>non-linear equations, 146, 150</td>
<td></td>
</tr>
<tr>
<td>normal distribution, 10</td>
<td></td>
</tr>
<tr>
<td>North America, 230, 239</td>
<td></td>
</tr>
<tr>
<td>Norway, 32</td>
<td></td>
</tr>
<tr>
<td>NPT, 28</td>
<td></td>
</tr>
<tr>
<td>null, 36, 40</td>
<td></td>
</tr>
<tr>
<td>obstacles, 237</td>
<td></td>
</tr>
<tr>
<td>oceans, 222</td>
<td></td>
</tr>
<tr>
<td>oil, 145</td>
<td></td>
</tr>
<tr>
<td>operating system, 81</td>
<td></td>
</tr>
</tbody>
</table>

Complimentary Contributor Copy
Index

110, 111, 142, 145, 161, 173, 202, 203, 205, 213, 214, 216
probability density function, 62, 145
probability distribution, 10, 63
procurement, 218
progesterone, 149, 157
programming, 6, 39, 101, 113, 126, 136, 153, 220
project, 56, 57, 59, 70
propane, 149, 157
prophase, 105, 107
protected areas, viii, 221, 222, 223, 224, 226, 238, 239
protection, 222, 223, 225, 226, 238
proteins, 27, 72

Q
quasi-equilibrium, 142

R
radio, 56
radius, 41
reactions, 147, 150
real time, 193
reality, 67
recombination, 25
reconstruction, 70, 94, 96, 97
regions of the world, 235
regulations, 224
rejection, 8, 31
relatives, 238
relaxation, 32
reliability, 56, 148, 150, 151, 160, 172, 174
repair, 193
requirements, viii, 6, 148, 221, 224
researchers, 100, 115
reserves, 222, 223, 239, 241
resonator, 180
resources, 56, 205, 222
response, 192
restoration, 32, 100
restrictions, 56, 57, 69, 217, 231
ribbonucleic acid, 72
risks, 222, 241
RNA, 72
root, 8, 73, 74, 77, 79, 150
routes, 193, 194, 205, 206
routines, 205
rules, 62, 102, 142, 146, 157, 194, 217

S
SA key components, vii
safety, 217
savings, 194, 215
scaling, 106, 119
school, 93
science, 8, 141, 238
scientific computing, 52
search space, vii, 8, 37, 40, 73, 74, 76, 77, 106, 107, 119
services, 219
shape, viii, 159, 160, 163, 173, 179, 222
showing, 61, 71, 75, 111
significance level, 85
silver, 238
simulation, vii, 67, 107, 111, 151, 170, 192, 194, 200, 209, 234
simultaneous phase, viii, 139, 140
Singapore, 152
software, 74, 209, 214, 215, 216, 217
solubility, 149, 156
solution space, 3, 4, 21, 107, 204, 205
solvents, 148
sound speed, 160
spacecraft trajectory, vii, 99, 100, 132
species, viii, 72, 74, 221, 223, 224, 225, 226, 227, 228, 230, 231, 232, 234, 235, 236, 238, 239, 241
species richness, 226, 232
specifications, 196, 224
spending, 192
Spring, 239
SSA, 6, 145
stability, viii, 14, 57, 58, 139, 140, 145, 146, 149, 153, 154, 155
stabilization, 102, 108
stakeholders, 237
standard deviation, 12, 47, 49, 62, 63, 85, 86, 93, 211
state(s), 5, 11, 15, 17, 29, 43, 73, 74, 75, 81, 86, 88, 92, 93, 102, 105, 124, 143, 145, 146, 152, 153, 157, 158, 173, 202, 203, 204
strategic planning, 192
strategy use, 105
stratification, 226
stress, 37, 51
stretching, 144, 145
structure, 78, 96, 143, 147, 195, 205, 213, 214, 216
substrate, 224
success rate, 122, 147, 149
Sun, 153
supplier, 192
supply chain, 192
surface area, 224
surrogates, 241
sustainability, 236
Sweden, 194, 219, 220
Switzerland, 240
synthesis, 148
Taiwan, 159
target, 128, 175, 213, 224
taxa, 72, 74, 76, 77, 85, 87, 223, 238
teams, 55
technical assistance, 237
techniques, 2, 8, 36, 143, 162, 209, 214, 217
technology, 217
teeth, 115
temperature, vii, 1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 15, 16, 17, 21, 22, 28, 30, 43, 56, 62, 63, 64, 65, 71, 76, 80, 81, 83, 84, 93, 102, 103, 104, 108, 142, 143, 146, 150, 161, 162, 173, 201, 202, 203, 204, 209, 210, 211, 213, 214, 215, 216, 217, 226, 227
termination rule, 102
testing, 36, 69, 70, 82, 83, 201, 213, 216
thermodynamic calculations, 139, 140, 141, 144, 145, 151
thermodynamics, viii, 141, 143, 146
tics, 71
time series, 222
topology, 73, 94
tracks, 200
trade, 80, 81, 84, 151, 237
trade-off, 151, 237
trajectory, vii, 99, 100, 117, 118, 119, 120, 121, 126, 132, 133, 137, 138
transformations, vii, 11, 71, 102, 145
transmission, viii, 159, 161, 163, 168, 178
transport costs, 195
transportation, 192, 193, 194, 195, 215, 217, 218, 219, 220
treatment, 51, 147, 230
trial, 120, 143, 208
trick drivers, 193
tunneling, 8, 31
turnover, 235

U

U.S. Department of Commerce, 241
UK, 238, 241
United States (USA), 70, 96, 191, 192, 218, 219, 224, 226, 227, 240, 241
updating, 51
US Department of Commerce, 240
US Sustainable Fisheries Act, viii

V

validation, 57
vapor, 148, 156
variables, vii, 8, 24, 26, 30, 31, 36, 37, 38, 40, 70, 107, 110, 112, 115, 119, 120, 125, 126, 140, 141, 147, 148, 150, 151, 195, 198, 200, 214, 216, 236
variations, 143

Complimentary Contributor Copy
vector, 7, 9, 38, 39, 40, 42, 44, 45, 99, 100, 101, 102, 119, 124, 126, 127, 130, 131, 133, 138, 144, 153
velocity, 118, 160, 162, 179, 183, 184
ventilation, 190
vessels, 237

W

Wales, 220
Washington, 153, 240, 241
water, 148, 149, 157

wave number, 160
windows, viii, 191, 193, 194, 197, 200, 215, 218
Wisconsin, 32
wood, viii, 191, 192, 193, 195, 196, 197, 198, 200, 204, 205, 206, 208, 209, 217, 218, 219, 220
wood products, 196

Y

yield, 84, 92, 181, 183