December 12, 2000
RT0390
Computer Science; Mathematics 20 pages

Research Report

A New Noise-Based Gradient Method and Its Applications

Hiroyuki Okano and Masato Koda

IBM Research, Tokyo Research Laboratory
IBM Japan, Ltd.
1623-14 Shimotsuruma, Yamato
Kanagawa 242-8502, Japan

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

A NEW NOISE-BASED GRADIENT METHOD AND ITS APPLICATIONS

Hiroyuki Okano* and Masato Koda†

* *IBM Research, Tokyo Research Laboratory*
1623-14 Shimotsuruma, Yamato, Kanagawa 242-8502 Japan
Tel: +81-46-215-4964
Fax: +81-46-273-7428
okanoh@jp.ibm.com

† *Institute of Policy and Planning Sciences*
University of Tsukuba
1-1-1 Tennoudai, Tsukuba Science City, 305-8573 Japan
Tel: +81-298-53-5222
Fax: +81-298-55-3849
koda@shako.sk.tsukuba.ac.jp

Abstract

A new gradient method applicable to local minimization of arbitrary functions is proposed. The proposed method, referred to as stochastic noise reaction (SNR), injects a Gaussian white noise sequence into each variable, and approximates a gradient by averaging the products of the objective function values and the injected noise. Because the gradient approximation in SNR only uses noise and objective function values, the target objective function need not be differentiable, and may even be arbitrary subroutines in a computer program. This property of SNR make it applicable in parametric optimization heuristics for discrete problems where an objective function is defined as a heuristic procedure that maps continuous parameters to a real value, and the continuous parameters are iteratively updated by a gradient-based method. Computational experiments using $tanh^2(x)$ show that SNR can locally minimize nonlinear functions starting from appropriate initial solutions. A Hopfield-Tank-type, a Held-Karp-type, and a novel addition heuristic-based formulations for the traveling salesman problem are introduced to discuss SNR's applicability to combinatorial optimization problems. Through intensive numerical experiments, SNR is shown to be applicable to these formulations.

Keywords

gradient method, combinatorial optimization, traveling salesman problem, stochastic sensitivity analysis, stochastic noise reaction, parametric optimization heuristics, addition heuristic

1. Introduction

Function minimization plays an important roll in various areas; such as, operations research, engineering, economics, etc. For functions defined by formulas, if they have certain properties, e.g., linear, quadratic, or twice-differentiable, there are a lot of mathematical optimization techniques available; e.g., linear programming, quadratic programming, gradient methods, the Newton method, etc. On the other hand, when an objective function is not defined by a formula, or it is defined by a non-differentiable or a discrete formula, the mathematical optimization techniques often fail to apply. Instead, heuristic approaches are more practical for such functions to obtain reasonably good approximate solutions. In this paper, a general gradient method which is applicable to arbitrary objective functions is proposed, and it is shown to be applicable in heuristic solutions for combinatorial optimization problems. The proposed method, referred to as *stochastic noise reaction (SNR)*, injects a Gaussian white noise sequence into each variable in the given objective function, and approximates a gradient (sensitivity), based on the stochastic sensitivity analysis, by averaging the products of the objective function values and the injected noise. The objective function can be non-differentiable or even exist only as subroutines in computer programs; that is because, in the gradient computation, SNR does not explicitly differentiate the objective function. This property of SNR may be able to fill the gap in solution approaches between mathematical programming and discrete heuristics. This study aims to present a new noise-based gradient method, as well as to move forward in the area of *parametric optimization heuristics* for discrete problems where an objective function is defined as a heuristic procedure that maps continuous parameters to a real value, and continuous parameters are iteratively updated by a gradient-based method. Throughout the paper, the *traveling salesman problem (TSP)* is used as an example of combinatorial optimization problems.

The TSP can be viewed as a problem of function minimization where the function maps a tour (or some parameters that indirectly specify a tour) to a real cost. In the TSP, a set of vertices $V = \{1, 2, \ldots, |V|\}$ and a distance between each pair of vertices i and j, d_{ij}, are given, and the problem seeks to find an ordering π of vertices that minimizes a tour length defined by

$$h(\pi) = \sum_{i=1}^{|V|} d_{\pi(i),\pi(i+1)}, \tag{1}$$

where the index of π is defined modulo $|V|$ so that vertex $\pi(|V|)$ is adjacent in the tour to both $\pi(|V|-1)$ and $\pi(1)$. The requirements that each vertex

should be visited exactly once and the ordering forms a connected tour are called the *tour constraints*. Note that here the geometric TSP is assumed, which means the vertices are mapped on a plane, and the distance is Euclidean. The objective function of the TSP, i.e., Eq. (1), takes a discrete vector π, and thus gradient-based methods, which are usually suitable for differentiable functions, cannot be applied directly. What are believed to be practical methods to obtain reasonably good approximate solutions are tour construction heuristics and local search. Construction heuristics implemented with k-d trees efficiently produce initial tours (solutions) [1]. The 2-opt heuristic, a type of local search, produces tours whose lengths are 4 to 5 percent in excess of optimal ones when it is applied to initial tours with particular properties [2]. The k-opt heuristic by Lin and Kernighan (LK) [3] produces tours whose lengths are 1 to 2 percent in excess of optimal, and without greatly depending on the initial tours. In local search, a set of neighborhood solutions of an ordering, $N(\pi)$, is defined, and the following transition is performed while there are improvements in the solution:

$$\pi \leftarrow \{\tilde{\pi} \mid \tilde{\pi} \in N(\pi),\ h(\tilde{\pi}) < h(\pi)\}.$$

For choosing a neighboring solution, $\tilde{\pi}$, a heuristic rule, such as $\min_{\tilde{\pi} \in N(\pi)} h(\tilde{\pi})$, is usually used because appropriate descent directions cannot be calculated mathematically.

Apart from the discrete local search approach in which algorithms refer to the discrete objective function, e.g., (1), there have been some attempts to relax discrete constraints and treat a combinatorial optimization problem as minimization of a continuous function. A linear programming (LP)-based approach (Dantzig et al. [5], Applegate et al. [6], etc.) formulates the TSP as an LP problem by introducing a real variable $x_{ij} \in [0, 1]$ for each edge (i, j), and iteratively adds subtour elimination constraints to the LP. An artificial neural network-based approach (Hopfield and Tank [7], Brandt et al. [8], etc.) formulates the TSP by introducing real variables $x_{Xi} \in [0, 1]$ where each value of x_{Xi} represents whether a vertex X is the i-th stop or not, and the tour constraint is taken into account as penalty terms in an objective function. The approach used by Held and Karp [9,10] can also be used as a TSP heuristic. In their formulation, a real variable $x_i \in R$ is introduced for each vertex i, and each distance d_{ij} is transformed to $d_{ij} + x_i + x_j$. Then a 1-tree that minimizes an objective function of x is obtained, where the 1-tree is a graph that consists of the minimum spanning tree (MST) for vertices $\{2, 3, \ldots, |V|\}$ and two distinct minimum cost edges from vertex 1 to the MST. The resulting 1-tree can be transformed to a tour, if it is not a feasible tour, by applying the

Christofides' algorithm [11]. In this paper, the artificial neural network-based (hereafter, called Hopfield-Tank (HT)-type, respecting the original work by Hopfield and Tank) and the Held-Karp (HK)-type formulations are used to compare deterministic (sub)gradient methods and SNR, a stochastic gradient method. Then a new formulation of the TSP, based on one of the construction heuristics, the *addition heuristic* [1,4], is proposed. The addition heuristic starts with a subtour of an arbitrary vertex, inserts vertices one by one to the subtour in the order of an *addition sequence*, such that the tour length is least increased by each addition. In the proposed formulation, a real priority $x_i \in R$ for each vertex i is introduced, and they are sorted in decreasing order of their values to generate an addition sequence. The priority vector x is iteratively updated by SNR so that the resulting tour length is locally minimized. Addition heuristics, such as the farthest addition, the nearest addition, or the random addition, have been believed to be worse than other better heuristics, such as the multiple fragment followed by the 2-opt [1,2]. The experiments in this paper, however, show that the qualities of solutions obtained by the addition heuristic-based approach are comparable to those by the Lin-Kernighan heuristic (LK) when an appropriate addition sequence is provided.

This paper is organized as follows: In Section 2, the algorithm of SNR is presented with numerical experiments to show its basic characteristics. In Section 3, SNR is compared to deterministic (sub)gradient methods by using HT- and HK-type formulations for the TSP. In Section 4, a novel TSP formulation based on the addition heuristic is proposed, and SNR is applied to it. Section 5 concludes this paper.

2. A Noise-Based Gradient Method

2.1. The Deterministic Gradient Algorithm

Consider an optimization problem

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in S, \end{array}$$

where a solution x is an n-dimensional column vector, and a feasible domain S is a set in $\mathbb{R}^n$. Gradient-based methods iteratively update the current solution x so that the objective function $f(x)$ is locally minimized as follows:

$$x \leftarrow x + \mu \frac{\delta x}{|\delta x|}, \quad \delta x = -\nabla f(x), \tag{2}$$

where μ is a step width, δx is a descent direction, and $\nabla f(x)$ is a gradient vector defined by

$$\nabla f(x) = \left(\frac{\partial f(x)}{\partial x_1}, \frac{\partial f(x)}{\partial x_2}, \ldots, \frac{\partial f(x)}{\partial x_n} \right)^T. \tag{3}$$

When $f(x)$ is convex, and it is non-differentiable at $\tilde{x}$, $\nabla f(\tilde{x})$ is replaced by a subgradient vector in $\partial f(x)$, a set of vectors $v \in \mathbb{R}^n$ such that $f(y) \geq f(x) + (y - x)^T v$. In this case, the algorithm (2) with $\delta x = -v$, $v \in \partial f(x)$, is called a subgradient method. The (sub)gradient-based methods (2) used with the algorithm framework described in Fig. 1, in this paper are called the *deterministic algorithms (DA)*. In the deterministic algorithms, Steps 6 to 11 in Fig. 1 are omitted, and δx in Steps 12 and 13 is deterministically computed by $-\nabla f(x)$ or by heuristics specific to the given problem.

2.2. Stochastic Noise Reaction

Unlike the deterministic algorithms that use partial differentiation or problem-specific heuristics to compute a descent direction, Koda and Okano used Gaussian white noise to approximate a gradient (sensitivity) of connection weights in artificial neural networks such that an error function for neural learning is locally minimized* [12]. They injected a Gaussian white noise sequence ξ_i into a state variable x_i for each neuron i, and derived an neural learning algorithm based on the stochastic sensitivity analysis [13] using the following identity (Novikov's Theorem):

$$E\left[\frac{\delta H(\xi)}{\delta \xi_i}\right] = E\Big[H(\xi)\xi_i\Big], \tag{4}$$

where $E[\cdot]$ denotes the expectation operator, $H(\xi)$ is an arbitrary function of Gaussian stochastic sequences ξ_i, $i = 1, 2, \ldots, n$, and $\frac{\delta H(\xi)}{\delta \xi_i}$ denotes the functional derivative [14]. Use of Gaussian white noise with zero mean and unit variance, denoted as $N(0, 1)$, is assumed; i.e.,

$$E[\xi_i] = 0, \quad E[\xi_i^t \xi_j^s] = \delta_{ij}\delta_{ts}, \tag{5}$$

where δ_{ij} and δ_{ts} denote the Kronecker delta, and ξ_i^t denotes the t-th noise in the noise sequence injected to the i-th variable.

*Koda and Okano called their neural learning algorithm *subconscious noise reaction* because their algorithm, which does not require deterministic back-propagation, is analogous to the subconscious activity of neurons.

Their approach can be generalized for gradient-based methods as follows: Inject a Gaussian white noise sequence ξ_i into a variable x_i, and denote the resulting stochastic variable and its instance by

$$x_i^* = x_i + \xi_i, \quad x_i^j = x_i + \xi_i^j. \tag{6}$$

Using the above Novikov's theorem, each component of a gradient is computed by

$$E\left[\frac{\partial f(x)}{\partial x_i}\right] = E\left[f(x^*)\xi_i\right] = \frac{1}{M}\sum_{j=1}^{M} f(x^j)\xi_i^j, \tag{7}$$

where the following identity has been used:

$$\frac{\partial f(x^*)}{\partial x_i} = \frac{\delta f(x^*)}{\delta \xi_i}. \tag{8}$$

Note that (8) is trivially derived from the following relationships:

$$\begin{array}{ll} \frac{\partial f(x^*)}{\partial x_i} = \frac{\partial f(x^*)}{\partial x_i^*}\frac{dx_i^*}{dx_i}, & \frac{dx_i^*}{dx_i} = 1, \\ \frac{\delta f(x^*)}{\delta \xi_i} = \frac{\partial f(x^*)}{\partial x_i^*}\frac{\delta x_i^*}{\delta \xi_i}, & \frac{\delta x_i^*}{\delta \xi_i} = 1. \end{array} \tag{9}$$

Note also that, in (7), all the components in a gradient $\nabla f(x)$, i.e., $\frac{\partial f(x)}{\partial x_i}$, $i = 1, 2, \ldots, n$, are computed at the same time, which means the gradient approximation requires the objective function to be evaluated M times. The value of M was set to 100 in all of the numerical experiments presented in this paper.

Using the gradient approximation (7) in the gradient method (2) yields a noise-based gradient method referred to as stochastic noise reaction (SNR). In this paper, SNR is performed within the algorithm framework described in Fig. 1. In Step 5 of the framework, a step width μ is gradually decreased from the predefined maximum value μ_{max} to the predefined minimum value μ_{min}. When μ_{max} and μ_{min} are set to the same value, the algorithm is called SNR/fixed. In the following sections, in SNR, $\mu_{min} = 0.1\mu_{max}$ is assumed. A noise sequence ξ_i for each variable x_i is formally generated in Step 8, while, in actual implementations, all the noise sequences should be generated and normalized in advance as described in the Appendix so that their mean values are guaranteed to be exactly zero. Before Step 12, $\delta x := \frac{\delta x}{M}$ is omitted because that averaging is canceled by the normalization in Step 13. Step 14 adjusts each variable x_i to a feasible domain S, where S is a hypercube defined by a value range $[x_{min}, x_{max}]$. When the problem is unconstrained, i.e., $S = \mathbb{R}^n$, then x_{max} and x_{min} are set to $+\infty$ and $-\infty$, respectively.

Algorithm framework of SNR

1. Initialize the current solution $x := x^0$.
2. Initialize the best solution $x^{best} := x$.
3. **For** $k := 1, 2, \ldots, N$ **do begin** // N iterations.
4. $a := (k-1)/(N-1)$. // $a \in [0 \to 1]$.
5. $\mu := (1-a)\mu_{max} + a\mu_{min}$. // $\mu \in [\mu_{max} \to \mu_{min}]$.
6. Initialize decent direction $\delta x := 0$.
7. **For** $j := 1, 2, \ldots, M$ **do begin** // Gradient approximation.
8. Generate a noise vector ξ^j. // $\xi_i^j \in N(0,1)$, $i = 1, 2, \ldots, n$.
9. $\delta x_i := \delta x_i - f(x^j)\xi_i^j$. // $E[f(x^*)\xi_i] = E[\frac{\partial f(x^*)}{\partial x_i}]$.
10. **If** $f(x^{best}) > f(x^j)$ **then** $x^{best} := x^j$. // Save the best solution.
11. **end;**
12. **If** $|\delta x| \simeq 0$ **then** $\delta x_i := \varepsilon \cdot \mathrm{sign}(\delta x_i)$. // Avoid a zero vector.[a]
13. $x := x + \mu \frac{\delta x}{|\delta x|}$. // Update the current solution.
14. $x_i := \max\{\min\{x_i, x_{max}\}, x_{min}\}$. // Adjust to the value range.
15. **If** $f(x^{best}) > f(x)$ **then** $x^{best} := x$. // Save the best solution.
16. **If** terminal condition is met **then goto** 18.
17. **end;**
18. Output x^{best}.

[a] The **then** part in Step 12 may be replaced by **goto** 18.

Figure 1. Algorithm framework of SNR.

2.3. Characteristics of SNR

SNR was applied to minimize $f(x) = tanh^2(x)$, which has a narrow *valley* around its global minimum at $x = 0$. The feasible domain was set to $S = R$. Figures 2 and 3 show a relationship between the initial value (solution) x^0 and the average of the snapshot values over 100 simulations of SNR/fixed and SNR, respectively, where the snapshot value means the current solution x after $N = 100$ iterations. In both Figs. 2 and 3, two data points for $\mu = 0.1$ and $\mu = 0.5$ are plotted together with the curve of $tanh^2(x)$. Figure 2 shows that, when μ was fixed at 0.1, the simulations starting from $x^0 \in [-10, +10]$ converged to $x = 0$ with high probability, while the simulations starting from $x^0 \in [-20, +20]$, when μ was fixed to 0.5, oscillated between $x = 0$ and $x = \pm 0.5$, and one of two results $f(0)$ or $f(\pm 0.5)$ was output depending on the initial values. Actually, the simulations also oscillated when $\mu = 0.1$, but the oscillation is not observed in the figure because $f(\pm 0.1) \simeq 0$.

The property of $\mu = 0.5$ such that a wider range of initial values are attracted to the global minimum compared to $\mu = 0.1$ is preferable, while the large oscillation is not desirable. Based on this observation, in order to make the

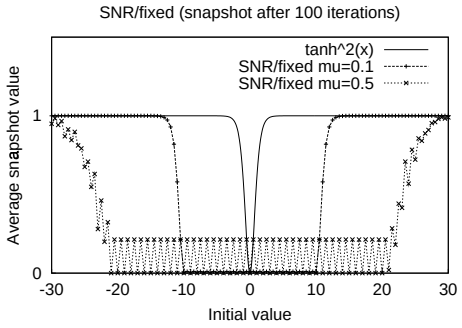

Figure 2. Initial and average snapshot values by SNR/fixed minimizing $tanh^2(x)$.

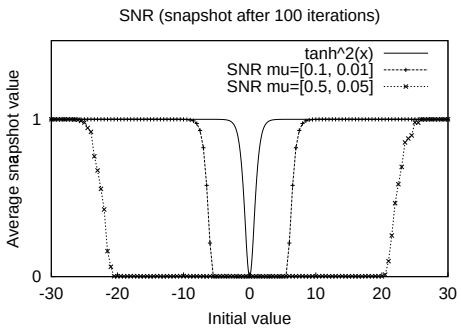

Figure 3. Initial and average snapshot values by SNR minimizing $tanh^2(x)$.

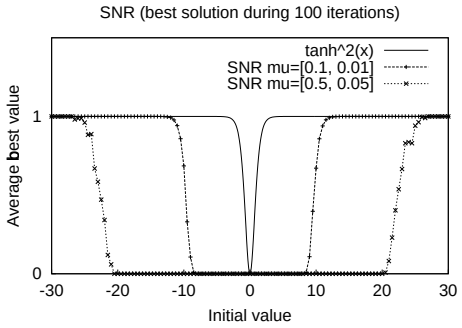

Figure 4. Initial and average best values by SNR minimizing $tanh^2(x)$.

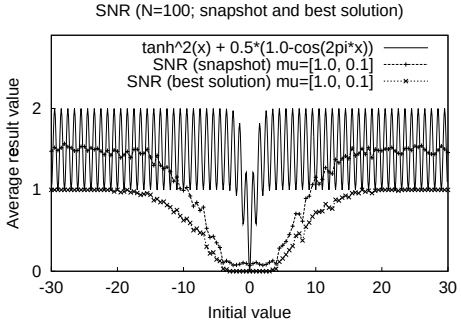

Figure 5. Initial and average result values by SNR minimizing $tanh^2(x) + \frac{1}{2}(1 - \cos(2\pi x))$.

attractive region wider and to avoid oscillation at the same time, μ is set to a large value and gradually decreased in SNR. In Fig. 3, for both settings of μ_{max}, no oscillation is observed. Note that, in Figs. 2 and 3, the displacements of the attractive regions from $x = 0$, when $\mu_{max} = 0.1$, are about 10 and 5.5, respectively, because of the longest displacement $\int_{k=1}^{N} \mu(k)$, where $\mu(k)$ denotes the value of μ at k-th iteration. When $\mu_{max} = 0.5$, on the other hand, they are not equal to their longest displacements of 50 and 27.5. This is because the gradient approximation of SNR fails to compute the correct descent direction (sign of the differential) when $\frac{df(x)}{dx} \simeq 0$. In Fig. 4, the initial value x^0 and the average of the best values over 100 simulations, each iterated for $N = 100$, are plotted for two settings of μ. The shapes of the plotted curves in Fig. 4 are similar to those in Fig. 3, except that the width of the attractive region

with $\mu_{max} = 0.1$ became wider than that in Fig. 3 for around the value of the displacement of $N(0, 1)$. This means that SNR may find good (or globally minimal) solutions in the loop of the gradient approximation (Steps 7 to 11).

In Fig. 5, the same simulation with $\mu_{max} = 1.0$ was conducted with an additional cosine curve to produce locally minimal solutions. The figure shows that SNR avoided small local minima and converged to the global minimum when it started from appropriate initial solutions. This is because small variations in the objective landscape are leveled and ignored in SNR's gradient approximation.

3. Comparison of SNR and DA

In this section, SNR is compared to deterministic algorithms (DA) using two TSP formulations: Hopfield-Tank (HT)-type and Held-Karp (HK)-type.

3.1. Experiments with the HT-type TSP formulation

3.1.1. Formulation

The example of a differentiable TSP formulation used here is based on that used by Hopfield and Tank [7]. The formulation denoted as TSP/HT is as follows:

$$\begin{aligned} f(x) &= \tfrac{A}{2} \textstyle\sum_X (\sum_i g(x_{Xi}) - 1)^2 \\ &+ \tfrac{B}{2} \textstyle\sum_i (\sum_X g(x_{Xi}) - 1)^2 \\ &+ C \textstyle\sum_X \sum_i g'(x_{Xi}) \\ &+ \tfrac{D}{2} \textstyle\sum_i \sum_X \sum_Y d_{XY} g(x_{Xi})(g(x_{Y,i-1}) + g(x_{Y,i+1})), \end{aligned} \tag{10}$$

where x_{Xi}, $X \in V$, $i = 1, 2, \ldots, |V|$, represents whether vertex X is the i-th stop in a tour, and $g(x)$ is a logistic function defined as

$$g(x) = \frac{1}{2}(1 + tanh(x)). \tag{11}$$

The number of variables n in this formulation is therefore $|V|^2$. The first term in the right hand side of (10) represents the tour constraint that each vertex should be visited exactly once, and the second term represents the constraint that exactly one vertex should be visited at the i-th stop. The third term increases the objective function value around the middle of the feasible domain, i.e., around $x_{Xi} = 0$, in order to accelerate the convergence of gradient-based algorithms. When the solution forms a feasible tour, the fourth

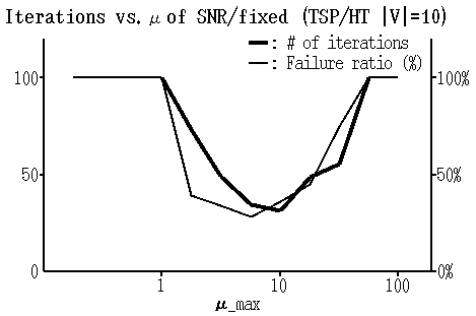

Figure 6. Average number of iterations and failure ratio of SNR/fixed for TSP/HT, plotted against μ_{max}.

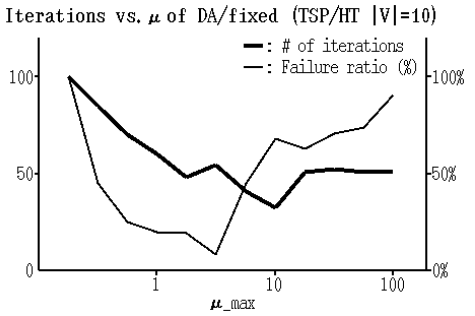

Figure 7. Average number of iterations and failure ratio of DA/fixed for TSP/HT, plotted against μ_{max}.

term, in which $g(x)$ are all replaced by x, computes the tour length. Note that, in (10), a logistic function is used to expand the feasible domain from a unit hypercube to an unconstrained domain, so that SNR can inject a noise into each variable in such a way that perturbed variables are still inside the feasible domain. A solution is regarded as a feasible tour if $\pi(i) = \{X \mid \max_{X \in V} x_{Xi}\}$, $i = 1, 2, \ldots, |V|$, form an ordering of vertices and $\pi(pos(X)) = X$ where $pos(X) = \{i \mid \max_i x_{Xi}\}$.

The objective function of TSP/HT is differentiable as follows:

$$\begin{aligned} \frac{\partial f(x)}{\partial x_{Zj}} &= Ag'(x_{Zj})(\textstyle\sum_i g(x_{Zi}) - 1) \\ &+ Bg'(x_{Zj})(\textstyle\sum_X g(x_{Xj}) - 1) \\ &+ Cg''(x_{Zj}) \\ &+ Dg'(x_{Zj}) \textstyle\sum_X d_{XZ}(g(x_{X,j-1}) + g(x_{X,j+1})). \end{aligned} \tag{12}$$

The value of $\delta x = -\nabla f(x)$ in DA is calculated accordingly.

3.1.2. Numerical experiments

In the numerical experiments, the parameters were set to a reasonably good setting, found by trial and error, to be $A = 5.0$, $B = 5.0$, $C = 1.0$, and $D = 4.0$. Ten instances, each consisting of $|V| = 10$ random vertices on a unit square were generated, and ten simulations from different initial solutions for each instance were performed, where initial solutions were set to small perturbations around zero: $x^0_{Xi} \in N(0, 1/4)$. In each simulation run, up to $N = 100$ iterations were performed. x_{max} and x_{min} were set to $+10$ and -10, respectively.

A simulation was considered to have converged and the iteration count, i.e., k in Fig. 1, was recorded when a feasible tour was first found during the

$N = 100$ iterations. If no feasible tour was found, the simulation was regarded as a failure. The failure ratio was defined as the ratio of failed simulations among 100 trials (10 runs each for 10 instances). Average numbers of iterations (for finding the first feasible tours) were measured over successful simulations. Figures 6 and 7 show the average numbers of iterations and the failure ratios of SNR/fixed and DA/fixed, respectively, applied to the TSP/HT formulation. In each simulation, μ was fixed at a single value because suppressing oscillation is not important in these experiments. The failure ratios of both SNR/fixed and DA/fixed form valley-shaped curves whose minimum points are around $\mu_{max} = 5.6$ and $\mu_{max} = 3.2$, respectively. The minimum failure ratios of SNR/fixed and DA/fixed were 28% and 8%, respectively, while their average numbers of iterations were 35 and 55, respectively.

It is to be noted that the performance of the stochastic method, SNR(/fixed), which does not compute an exact gradient by differentiation, is comparable to that of the deterministic algorithm when μ is set to appropriate values. Note also that each component $\frac{\partial f(x)}{\partial x_{Xi}}$ of a gradient for $X \in V$, $i = 1, 2, \ldots, |V|$, that is $|V|^2 = 100$ values, were approximated at the same time by $M = 100$ evaluations of the objective function. This means that each gradient is approximated *concurrently* during M iterations of Steps 7 to 11 (Fig. 1). This concurrent approximation is achieved because Gaussian white noise sequences are not correlated with each other; i.e., (5).

Tours obtained by both algorithms were not always optimal. When μ was large, simulations tended to converge to bad local optima. Note also that the HT-based TSP formulation is not considered to be practical, and it is only used here for the evaluation of SNR.

3.2. Experiments with the HK-type TSP formulation

3.2.1. Formulation

Held and Karp proposed in [9,10] a TSP formulation in which the tour constraint is relaxed to represent a solution by a graph called a 1-tree (Fig. 8 (a)), which consists of the minimum spanning tree (MST) for vertices $\{2, 3, \ldots, |V|\}$ and two distinct minimum cost edges from vertex 1 to the MST. In their formulation, a real variable $x_i \in R$ is introduced for each vertex i, and each distance is transformed as:

$$d_{ij} \rightarrow d_{ij} + x_i + x_j. \tag{13}$$

This transformation changes the minimum cost 1-tree, while the minimum cost tour is left unchanged.

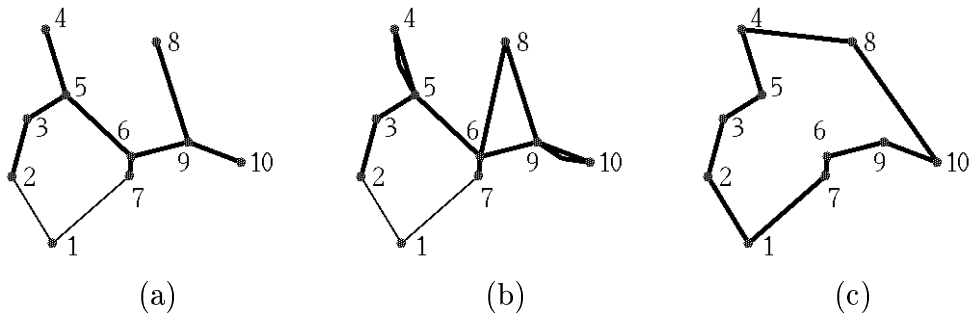

Figure 8. An example of 1-tree and a postprocess to obtain a feasible tour. (a) A 1-tree. (b) A 1-tree along with the minimum cost matching for odd degree vertices. (c) A feasible tour.

The objective function is defined as follows:

$$f(x) = L + 2\sum_i x_i - onetree(x), \tag{14}$$

where L is a constant value, and $onetree(x)$ denotes the cost of the minimum cost 1-tree with respect to the distances transformed by x. In numerical experiments in this section, L is set to the cost of the LK tour so that $f(x) \geq 0$. In minimizing (14), the solution x is updated in such a way that the degree of each vertex i, $deg(x,i)$ approaches 2, and the resulting 1-tree finally forms a feasible tour. Note that a feasible tour is also a 1-tree in which each vertex has degree 2. When x_i is increased by a small value dx_i, if the corresponding 1-tree does not change, the increase in $f(x)$ is $dx_i(2-deg(x,i))$. A subgradient vector v for use in DA can therefore be calculated as

$$v_i = -deg(x,i). \tag{15}$$

Note that a constant variation in the subgradient vector does not affect the value of (14) because $2\sum_i c = \sum_i c \cdot deg(x,i) = 2Nc$.

Using Held and Karp's formulation does not guarantee obtaining a feasible tour. In this paper, the Held and Karp's formulation is used with the post-process in the Christofides' algorithm [11] to guarantee always obtaining a feasible tour, and it is denoted as TSP/HK formulation. In the TSP/HK formulation, a feasible tour (an ordering of vertices π) with respect to a solution vector x is obtained by the procedure $CH(x)$ defined as follows:

Procedure $CH(x)$

1. Obtain the minimum cost 1-tree with respect to the distance transformed by the solution vector x (Fig. 8 (a)).

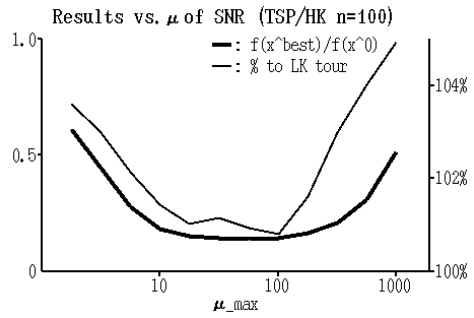

Figure 9. Average result value and tour length of SNR for TSP/HK, plotted against μ_{max}.

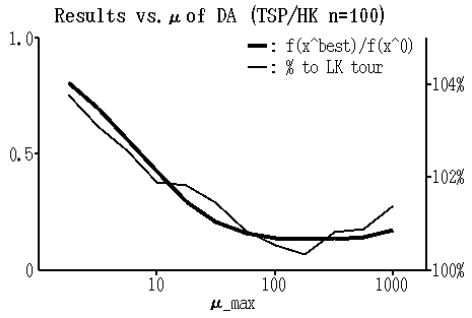

Figure 10. Average result value and tour length of DA for TSP/HK, plotted against μ_{max}.

2. Add the minimum cost matching for odd degree vertices to the 1-tree with respect to the original distance (Fig. 8 (b)).
3. Shortcut a pair of edges connected to vertex i, $deg(x,i) \geq 4$, such that the longest edge among those connected to vertex i is removed, the tour length with respect to the original distance is most decreased, and the graph remains connected.
4. If there are vertices whose degrees are four or more, go to 3.
5. Output $\pi(i)$, $i = 1, 2, \ldots, |V|$, the ordering of vertices in the resulting tour (Fig. 8 (c)).

Note that the comparison of the objective function values does not match that of the tour lengths as defined by Eq. (1); which means, $f(x) > f(y)$ does not necessarily guarantee $h(CH(x)) > h(CH(y))$. Note also that a 1-tree and the minimum cost matching are both computed in polynomial time.

3.2.2. Numerical experiments

Ten instances, each consisting of $|V| = 100(= n)$ random vertices on a unit square were generated, and SNR and DA were applied. Each simulation executed for $N = 100$ iterations, and $f(x^{best})/f(x^0)$ was evaluated. Both the best solution with respect to $f(x)$ and the shortest tour x^{tour} found during a simulation were recorded. Note that x^{tour} is not necessarily the same as x^{best}. In Fig. 9, the average values of $f(x^{best})/f(x^0)$ and $h(CH(x^{tour}))/LK$ obtained by SNR for ten instances are plotted against μ_{max}, where LK denotes the length of the tour obtained by LK. The values of $f(x^{best})/f(x^0)$ and $h(CH(x^{tour}))/LK$ were minimized at $\mu_{max} = 56$ and $\mu_{max} = 100$, respectively. The minimum value of $h(CH(x^{tour}))$ was 0.8% in excess of the LK tours on the average, which is about 2.3% in excess of the optimal tours. Figure 10 shows

the result by DA. When $\mu_{max} = 178$, $f(x^{best})/f(x^0)$ and $h(CH(x^{tour}))/LK$ were both minimized, and the obtained tour was 0.3% in excess of the LK tours on the average, which is about 1.8% in excess of the optimal tours.

Comparison of the results by SNR and DA (Figs. 9 and 10) shows that SNR's performance in minimizing the HK-type objective function is comparable to that of DA. Note that the important point which is focused on here is that a method which does not reference differential or heuristic subgradient information can minimize the TSP function in the TSP/HK formulation, and its performance is comparable to the deterministic approach. The processing times required by SNR, DA, and LK are 65, 1.5, and 0.15 seconds, respectively, on an IBM RS/6000 with a PowerPC 601 112-MHz CPU. Which means that the TSP/HK formulation, as well as the TSP/HT, is not better than the best known TSP heuristic, i.e., LK, as a heuristic for obtaining feasible tours. Note that the HK-type formulation is still useful for obtaining lower bounds of tour lengths.

4. The Addition Heuristic-Based TSP Formulation

In this section, a novel TSP formulation based on the addition heuristic is proposed, and SNR is applied to it. Because the authors do not know of any algorithm to compute a subgradient for this formulation deterministically, DA is not used in this section.

4.1. The Addition Heuristic

The addition heuristic, starting from a subtour consisting of a single vertex, inserts vertices one by one into the place in the subtour that least increases the tour length. An ordering of vertices, $a(i)$, $i = 1, 2, \ldots, |V|$, to insert into the subtour is called addition sequence. The offline addition heuristic, to which a is given in advance, is defined as follows:

Offline addition heuristic $AH(a)$

1. Let $a(1)$ be a subtour consisting of a single vertex.
2. **For** $i = 2, 3, \ldots, |V|$ **do begin**
3. Let s be the nearest point in the subtour from $a(i)$ (Fig. 11 (a)).
4. Let $W(s, a(i))$ be a set of vertices in the subtour inside the circle of radius $2d_{s,a(i)}$ centered at $a(i)$ (Fig. 11 (b)).
5. Find the place in the subtour, either before or after $t \in W(s, a(i))$, that least increases the tour length (Fig. 11 (c)).

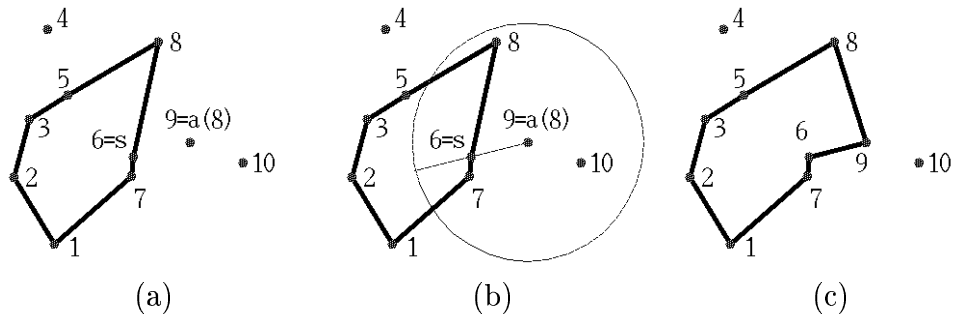

Figure 11. An example of the insertion procedure in the addition heuristic. (a) A subtour. The next vertex to be inserted into the subtour is 9. (b) Candidates for insertion positions – $(1,7)$, $(7,6)$, $(6,8)$, and $(8,5)$ – are identified. (c) Vertex 9 is inserted between vertices 6 and 8.

6. **end;**
7. Output $\pi(i)$, $i = 1, 2, \ldots, |V|$, the ordering of vertices in the resulting tour.

When a is a random ordering, the procedure is called the *random addition (RA)*. Note that the nearest vertex search and the fixed-radius near-neighbor search in Steps 3 and 4 are efficiently performed by using a k-d tree.

In the addition heuristic, local changes in the input addition sequence a do not greatly change the resulting tour. For example, when a subtour consists of $|V| - 2$ vertices, insertions of $a(|V| - 1)$ and $a(|V|)$ to the subtour can be performed independently in most cases. Orders (or priorities) of the vertices in a, on the other hand, do affect the quality of the resulting tour. Misono and Iwano observed that the addition heuristic generates good tours when each subtour shares the regional characteristics of the optimal tour, and called such a subtour a *grand tour* [15]. According to their observation, it is conjectured that a good tour can be obtained if key vertices which characterize the optimal tour have higher priorities in the addition sequence.

4.2. Addition Heuristic-Based Continuous Function

The addition heuristic $AH(a)$ described in the last subsection cannot be used directly in (sub)gradient-based methods because it takes a discrete vector a. To relax the discrete property, an n-dimensional real vector x is introduced where $n = |V|$. Each component x_i specifies a priority of vertex i in an addition sequence; i.e., an addition sequence is generated by sorting x_i in decreasing order so that $x_{a(i)} \geq x_{a(i+1)}$. Then, the formulation denoted as

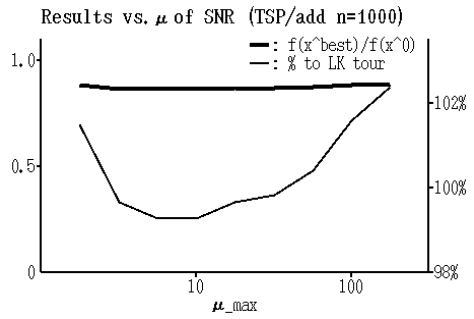

Figure 12. Average result values of SNR for TSP/add, plotted against μ_{max}.

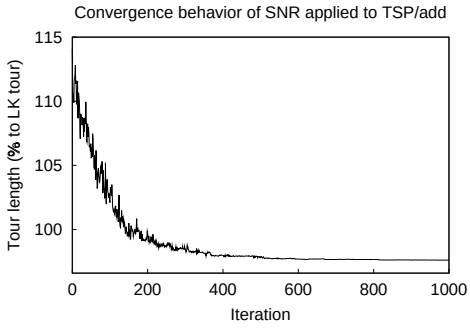

Figure 13. Convergence behavior of SNR applied to TSP/add.

TSP/add is defined as follows:

$$f(x) = h(AH(decreasingorder(x))), \tag{16}$$

where $decreasingorder(x)$ maps the priority vector x to an addition sequence a, $AH(a)$ generates an ordering π by using the addition heuristic, and $h(\pi)$, i.e., (1), computes a corresponding tour length.

The objective function of the TSP/add formulation (16) is normally defined as a subroutine in a computer program. Therefore, it is not differentiable, and gradient-based deterministic algorithms cannot be applicable. On the other hand, SNR does not need to explicitly differentiate the objective function, and thus SNR is applicable to the TSP/add formulation.

4.3. Numerical Experiments

Ten random instances on a unit square were generated and used as input for SNR, RA, a random local search, and LK. The size of each instance is $|V| = 1000$, and the maximum iterations in each simulation was set to $N = 1000$. Figure 12 shows the average values of $f(x^{best})/f(x^0)$ and $f(x^{best})/LK$ for the ten instances plotted against μ_{max}. The objective function was minimized at $\mu_{max} = 10$, and the value was 99.3% of the LK tours on the average, which is within 1% of the optimal tours. Figure 13 shows the convergence behavior of SNR with $\mu_{max} = 10$ applied to one of the instances. It is observed that the solution goes up and down to a large extent during the first 400 iterations, and gradually converges to a local minimum solution. The components in the local optimum solution x are distributed in $[-20, +30]$. The processing times required by SNR and LK are 2808 and 6.4 seconds, respectively, on an IBM

RS/6000 with a PowerPC 601 112-MHz CPU.

RA was executed $M \times N = 100,000$ times with different randomly generated addition sequences, and the best values obtained during the $M \times N$ trials was recorded. A local search with a random neighborhood $N(x)$ was also executed $M \times N$ times where each component in a neighborhood solution $\tilde{x} \in N(x)$ was generated randomly by $\tilde{x}_i = x_i + N(0,1)$. Tours generated by RA and the random local search were 109% and 108%, respectively, of the LK tours on the average over the ten random instances; which means the performances of both random trial and random descending are much worse than the gradient-based approach. Note that, in each simulation of SNR, the AH procedure is called $M \times N$ times.

The above experiments show that the proposed TSP/add formulation (16) has a *quasi-continuous* (non-differentiable but smooth) landscape in the sense that stochastic gradient methods can find a locally optimal solution by an iterative procedure. While deterministic gradient methods are not applicable to this type of non-differentiable function (or heuristic procedure implemented as a computer program), SNR could be applied without much effort and could locally minimize the target function. Note that the presented experiments show the possibility of addition heuristic-based methods which generate tours comparable to LK, the best known TSP heuristic.

5. Conclusion

A new noise-based gradient method referred to as SNR was proposed and evaluated with three TSP formulations: the well-known Hopfield and Tank (HK)-type and Held and Karp (HT)-type formulations, and a novel addition heuristic-based formulation. Numerical experiments using HK- and HT-type TSP formulations showed that SNR has basically the same performance characteristics as deterministic algorithms based on gradient or subgradient methods. In the course of the evaluations, a new TSP formulation based on the addition heuristic was proposed. SNR was applied to the addition heuristic-based TSP formulation, and shown to be able to locally minimize the objective function defined by the presented formulation.

Through intensive numerical experiments, it has been shown that the described noise-based gradient method, SNR, has the following characteristics:

- It is applicable to arbitrary functions whenever their objective landscapes are quasi-continuous (smooth in a stochastic sense).
- It is applicable to non-differential objective functions, such as those defined as subroutines in computer programs, even if their gradient or subgradient

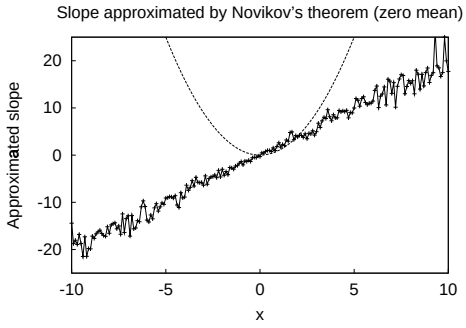

Figure 14. Slope of x^2 approximated with a noise sequence whose mean is exactly zero.

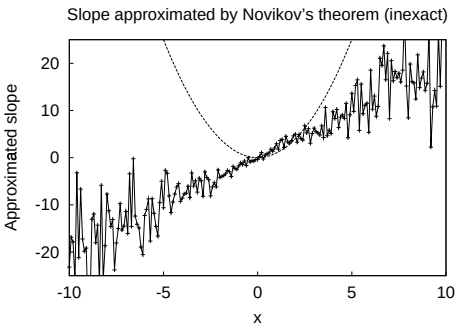

Figure 15. Slope of x^2 approximated with a noise sequence whose mean is inexact.

vectors cannot be obtained analytically.

- Small variations (local optima) in an objective landscape are leveled and ignored.

The above properties of SNR fill the gap in solution approaches between discrete heuristics and mathematical programming, and make it applicable in combinatorial optimization algorithms. For example, the addition heuristic-based TSP formulation (TSP/add) proposed in this paper takes continuous parameters and generates a discrete solution by using a heuristic procedure. The solution of the TSP/add using SNR can be viewed as a parametric optimization heuristic coupled with a noise-based gradient method, a stochastic mathematical programming algorithm. To the authors' knowledge the paradigm shown in this paper, i.e., application of a gradient method in parametric optimization heuristics for discrete problems, has not previously been focused. Future work will include applications of SNR for other parametric optimization heuristics for combinatorial optimization problems.

Appendix

In actual implementations of SNR, the mean values of the Gaussian white noise sequences $\xi_i \in N(0,1)$ should satisfy be exactly zero. To ensure this, after generating noise values as

$$\xi_i^j := \cos(2\pi a)\sqrt{-2\log b},\ i = 1, 2, \ldots, n,\ j = 1, 2, \ldots, M, \tag{17}$$

each noise value is normalized as

$$\xi_i^j := \xi_i^j - \frac{1}{M}\sum_{j=1}^{M} \xi_i^j, \tag{18}$$

where a and b are real numbers uniformly distributed on the $(0, 1)$-interval. (0 and 1 are not included.) The mean of the resulting noise sequence is exactly zero; i.e., $E[\xi_i] = 0$. Figure 14 shows an approximated slope of $f(x) = x^2$ using the normalized noise sequence, and Fig. 15 shows one using an inexact noise sequence. In both simulations, M was set to 100. Comparison of the two figures shows that normalizing the noise sequence to have a zero mean is important when the absolute value of x is large. Because the computation of Eq. (17) is expensive, in the numerical experiments, the same set of noise sequences was used for each iteration (Steps 3 to 17 in Fig. 1). The set of normalized noise sequences were generated in the initialization phase, and variables x_i, $i = 1, 2, \ldots, n$, and the n noise sequences are randomly matched in each iteration.

Note that the requirement for the variance of the noise to be 1 is not important, although it is a necessary condition for Novikov's theorem in Eq. (4), because the approximated gradient is used with a step width μ.

References

[1] J. L. Bentley, "Fast Algorithms for Geometric Traveling Salesman Problems," *ORSA Journal on Computing*, vol. 4, pp. 387-411, 1992.

[2] H. Okano, S. Misono, and K. Iwano, "New TSP Construction Heuristics and Their Relationships to the 2-Opt," *Journal of Heuristics*, vol. 5, pp. 71-88, 1999.

[3] S. Lin and B.W. Kernighan, "An Effective Heuristic Algorithm for the Traveling-Salesman Problem," *Operations Research*, vol. 21, pp. 498-516, 1973.

[4] G. Reinelt, *The Traveling Salesman, Computational Solutions for TSP Applications.* Heidelberg: Springer-Verlag, 1994.

[5] G. Dantzig, R. Fulkerson, and S. Johnson, "Solution of a Large-Scale Traveling-Salesman Problem," *Journal of Operations Research Society of America*, vol. 2, pp. 393-410, 1954.

[6] D. Applegate, R. Bixby, V. Chvátal, and W. Cook, "On the Solution of Traveling Salesman Problems," *Proc. of the International Congress of Mathematicians*, Doc. Math, pp. 645-656, 1998.

[7] J. J. Hopfield and D. W. Tank, "Neural Computation of Decisions in Optimization Problems," *Biological Cybernetics*, vol. 52, pp. 141-152, 1985,

[8] R. D. Brandt, Y. Wang, A. J. Laub, and S. K. Mitra, "Alternative Networks for Solving the Traveling Salesman Problem and the List-Matching Problem," *Proc. of IEEE International Conference on Neural Networks*, vol. II, pp. 333-340, 1988.

[9] M. Held and R. M. Karp, "The Traveling-Salesman Problem and Minimum Spanning Trees," *Operations Research*, vol. 18, pp. 1138-1162, 1970.

[10] M. Held and R. M. Karp, "The Traveling-Salesman Problem and Minimum Spanning Trees: Part II," *Mathematical Programming*, vol. 1, pp. 6-25, 1971.

[11] N. Christofides, "Worst-Case Analysis of a New Heuristic for the Traveling Salesman Problem," *Report 388*, Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh, PA, 1976.

[12] M. Koda and H. Okano, "A New Stochastic Learning Algorithm for Neural Networks," *Journal of Operations Research Society of Japan*, vol. 43, pp. 469-485, 2000.

[13] M. Koda, "Neural Network Learning Based On Stochastic Sensitivity Analysis," *IEEE Transactions on Systems, Man, and Cybernetics Part B*, vol. 27, pp. 132-135, 1997.

[14] D. K. Dacol and H. Rabitz, "Sensitivity Analysis of Stochastic Kinetic Models," *Journal of Mathematical Physics*, vol. 25, pp. 2716-2727, 1984.

[15] S. Misono and K. Iwano, "Experiments on TSP Real Instances," *IBM Research Report*, RT0153, 1996.