14

Evaluating the α-Dominance Operator in Multiobjective Optimization for the Probabilistic Traveling Salesman Problem with Profits

Bingchun Zhu[1], Junichi Suzuki[2] and Pruet Boonma[3]
[1,2]*University of Massachusetts, Boston*
[3]*Chiang Mai University*
[1,2]*USA*
[3]*Thailand*

1. Introduction

This chapter investigates a noise-aware dominance operator for evolutionary multiobjective optimization algorithms (EMOAs). An EMOA uses a population of individuals, each of which represents a solution candidate. It evolves individuals through generations and seeks the optimal solution(s) in a multiobjective optimization problem (MOP), which is formalized as follows.

$$\left.\begin{array}{r} \min\ F(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), \cdots, f_m(\vec{x})]^T \in \mathcal{O} \\ \text{subject to } \vec{x} = [x_1, x_2, \cdots, x_n]^T \in \mathcal{S} \end{array}\right\} \quad (1)$$

$\mathcal{S}$ denotes the decision variable space. $\vec{x}$ denotes a decision variable vector (or solution candidate) with respect to $\mathcal{S}$. It is called an individual in EMOAs. A function vector, $F : \mathbb{R}^n \rightarrow \mathbb{R}^m$, consists of m real-value objective functions, each of which produces an objective value with respect to the objective space $\mathcal{O}$. An MOP is to find an individual(s) that minimizes objective values with subject to $\mathcal{O}$.

In an MOP, objective functions (i.e., $f_1(\vec{x}), \cdots, f_m(\vec{x})$ in Equation 1) often conflict with each other; there exist rarely a single individual that is optimum with respect to all objectives. Therefore, an MOP often aims to find the optimal trade-off solutions, or Pareto-optimal solutions, by balancing conflicting objectives simultaneously. The notion of *dominance* plays an important role to seek Pareto optimality in MOPs (Srinivas & Deb, 1995). An individual $\vec{x} \in \mathcal{S}$ is said to *dominate* an individual $\vec{y} \in \mathcal{S}$ (denoted by $\vec{x} \succ \vec{y}$) iif the both of the following conditions are hold.

- $f_i(\vec{x}) \leq f_i(\vec{y}) \ \forall\ i\ = 1, \cdots, m$
- $f_i(\vec{x}) < f_i(\vec{y}) \ \exists\ i\ =\ 1, \cdots, m$

In real-world MOPs, objective functions tend to contain noise (Beyer, 2000; Bianchi et al., 2009). Thus, objective functions can yield different objective values from the same individual from time to time. For considering this noise, Equation 1 is revised as follows.

$$\left.\begin{array}{r} \min\ F(\vec{x}) = [f_1(\vec{x}) + \epsilon_1, f_2(\vec{x}) + \epsilon_2, \cdots, f_m(\vec{x}) + \epsilon_m]^T \in \mathcal{O} \\ \text{subject to } \vec{x} = [x_1, x_2, \cdots, x_n]^T \in \mathcal{S} \end{array}\right\} \quad (2)$$

ϵ_m represents noise in the m-th objective function. Noise in objective functions can interfere with a dominance operator, which determines dominance relationships among individuals. For example, a dominance operator may mistakenly judge that an inferior individual dominates an superior one. Defects in a dominance operator significantly degrades the performance (e.g., convergence velocity) to solve MOPs (Arnold, 2000; Beyer, 2000; Beyer & Sendhoff, 2007; Bianchi et al., 2009; Carroll et al., 2006; Diwekar & Kalagnanam, 1997).

In order to address this issue, this chapter proposes a notion of noise-aware dominance, called *α-dominance*, and studies the α-dominance operator for EMOAs. This operator takes objective value samples of given two individuals, estimates the impacts of noise on the samples and determines whether it is confident enough to judge a dominance relationship between the two individuals. Unlike existing noise-aware dominance operators, the α-dominance operator assume no noise distributions a priori. (See Section 5. for more details.) Thus, it is well applicable to a variety of real-world MOPs whose objective functions follow unknown noise distributions.

This chapter describes the design of the α-dominance operator and evaluates it with the probabilistic traveling salesman problem with profits (pTSPP), which can derive a number of real-world noisy MOPs. pTSPP is a combination of existing two variants of the traveling salesman problem (TSP): the probabilistic TSP (pTSP) (Jaillet, 1985) and the TSP with profits (TSPP) (Feillet et al., 2005). In experimental evaluation, the α-dominance operator is integrated with NSGA-II (Deb et al., 2000), a well-known EMOA, and compared with existing noise-aware dominance operators. Experimental results demonstrate that the α-dominance operator reliably performs dominance operation in pTSPP and outperforms existing noise-aware operators in terms of the optimality, convergence velocity and diversity of individuals.

The remaining part of this chapter is structured as follows. A further related work, particularly in the areas of probabilistic Traveling Salesman Problems(TSPP) and noise handling techniques of evolutionary algorithms, is surveyed in Section 5. Section 2.proposes a new problem, probabilistic Traveling Salesman Problem with Profit (pTSPP) and describes its objectives and objective function. Section 3.introduces background of MOEAs and describes a variant of NSGA-II, a well-known EMOA. To handling the uncertainties whose distribution is unknown a priori, a noise-aware dominance operator is proposed in Section **??**. This section also presents the integration of proposed noise-aware dominance operator and NSGA-II. Then, Section **??** reports computational results of proposed noise-aware dominance operator on some test pTSPP problems with comparison to some other noise-aware dominance operators. Section 6.concludes this research with a summary.

2. Probabilistic Traveling Salesman Problem with Profits (pTSPP)

This paper uses the following notations to define pTSPP. pTSPP is defined on a fully-connected graph $G = (V, E)$.

- $V = \{v_0, v_1, v_2, ..., v_n\}$ is a set of vertices in G, where v_0 is the depot. $V' = V - \{V_0\}$ is a set of n vertices. This paper assumes that vertices are stationary, and $|V|$ does not change dynamically.

- $E = \{v_i, v_j | v_i, v_j \in V; i \neq j\}$ is the set of edges. Each edge $\{v_i, v_j\} \in E$ has an associated cost c_{v_i,v_j}.
- Each vertex $v_i \in V'$ maintains a visiting probability p_{v_i}, which represents the probability that v_i is visited. $p_{v_i} \in [0.0, 1.0]$. The visiting probability of the depot $p_{v_0} = 1.0$.
- Each vertex $v_i \in V'$ has an associated profit $\rho_{v_i} \geq 0.0$. The depot's profit $\rho_{v_0} = 0.0$.
- R is a sequence of vertices, starting and ending with v_0. R may not contain all the vertices in V': $|R| \leq |V'| + 2$. No redundant vertices exist in R. (A node is never visited more than once.) R is an *a posteriori* route; the salesman uses it to decide a posteriori which vertices he actually visits based on the visiting probabilities associated with vertices in R.

pTSPP is to find the Pareto-optimal routes with respect to the following two objectives.

- *Cost*: The total traveling cost that the salesman incurs by visiting vertices in a route. This objective is to be minimized. It is computed as:

$$f_{cost} = \sum_{v_n, v_{n'} \in R} p_{v_n} p_{v_{n'}} c_{v_n, v_{n'}} \tag{3}$$

  where $v_{n'}$ is the next vertex of v_n in R.
- *Profit*: The total profit that the salesman gains by visiting vertices in a route. This objective is to be maximized. It is computed as:

$$f_{profit} = \sum_{v_n \in R} p_{v_n} \rho_{v_n} \tag{4}$$

Two objectives in pTSPP conflict with each other. For example, a shorter route (i.e., a route containing a smaller number of vertices) yields a lower cost and a lower profit. On the contrary, a longer route (i.e., a route containing a larger number of vertices) yields a higher cost and a higher profit.
pTSPP inherently considers noise in its objective functions. Following the notations in Equation 2, pTSPP is formulated as follows.

$$\left.\begin{aligned} \min\ F(R) = [f_{cost}(R) + \epsilon_{cost}, \tfrac{1}{f_{profit}(R)} + \epsilon_{profit}]^T \in \mathcal{O} \\ \text{subject to } R = [v_0, \cdots, v_n, v_{n'} \cdots, v_0] \in \mathcal{S} \end{aligned}\right\} \tag{5}$$

As mentioned in Section 1, pTSPP is a combination of pTSP (Bertsimas & Howell, 1993; Jaillet, 1985) and TSPP (Feillet et al., 2005). pTSP is to find an optimal *a priori* route with the minimum cost in which each vertex requires a visit of the salesman with a given visiting probability. TSPP is to find the optimal route, with respect to profit as well as cost, with which the salesman visit a subset of given vertices. pTSPP extends pTSP in a sense that pTSPP computes the total profit of a route based on the profit and visiting probability associated with each vertex in the route (Equation 4). Unlike TSPP, pTSPP considers a visiting probability for each vertex.
A number of real-world noisy MOPs can be reduced to pTSPP as various real-world optimization problems can be reduced to pTSP and TSPP (Bertsimas & Howell, 1993; Feillet et al., 2005; Jaillet, 1985; Jozefowiez et al., 2008a). For example, pTSPP can represent noisy MOPs in transportation planning, supply chain networks, data routing/gathering in computer networks.

3. The proposed evolutionary multiobjective optimization algorithm for pTSPP

This section describes the proposed noise-aware evolutionary multiobjective optimization algorithm (EMOA) to solve pTSPP. It is designed to evolve individuals (i.e., solution candidates) toward the Pareto-optima through generations with various operators such as parent selection, crossover, mutation, selection, individual ranking and diversity preservation operators. The α-dominance operator is used in the parent selection and individual ranking operators. Section 3.1explains the representation of individuals in the proposed algorithm. Section 3.2overviews the algorithmic structure of the proposed algorithm. Sections 3.3to 3.5 describe key operators in the proposed algorithm.

3.1 Individual representation

In the proposed EMOA, each individual represents a solution candidate for pTSPP: an a posteriori route R that contains a sequence of vertices. (See Section 2.) Every individual has the depot (v_0) as its first and last element. Figure 1 shows an example individual. Given this route, the salesman starts with v_0, visits v_3 and its subsequent 7 nodes, and returns back to v_0.

0	3	5	7	2	10	4	9	1	0

Fig. 1. The Structure of an Example Individual

Different individuals have different lengths, depending on the number of nodes to be visited.

3.2 Algorithmic structure

Listing 1 shows the algorithmic structure of evolutionary optimization in the proposed EMOA. It follows the evolutionary optimization process in NSGA-II, a well-known existing EMOA (Deb et al., 2000).

At the *0*-th generation, N individuals are randomly generated as the initial population $\mathcal{P}_0$ (Line 2). Each of them contains randomly-selected vertices in a random order. At each generation (g), a pair of individuals, called parents (p1 and p2), are chosen from the current population $\mathcal{P}_g$ using a binary tournament (Lines 6 and 7). A binary tournament randomly takes two individuals from $\mathcal{P}_g$, compares them based on the α-dominance relationship between them, and chooses a superior one as a parent.

With the crossover rate P_c, two parents reproduce two offspring with a crossover operator (Lines 8 and 9). Each offspring performs mutation with the mutation rate P_m (Lines 10 to 15). The binary tournament, crossover and mutation operators are executed repeatedly on $\mathcal{P}_g$ to reproduce N offspring. The offspring ($\mathcal{O}_g$) are combined with the parent population $\mathcal{P}_g$ to form $\mathcal{R}_g$ (Line 19).

The selection process follows the reproduction process. N individuals are selected from $2N$ individuals in $\mathcal{R}_g$ as the next generation's population ($\mathcal{P}_{g+1}$). First, the individuals in $\mathcal{R}_g$ are ranked based on their α-dominance relationships. Non-dominated individuals are on the first rank. The i-th rank consists of the individuals dominated only by the individuals on the $(i-1)$-th rank. Ranked individuals are stored in $\mathcal{F}$ (Line 20). $\mathcal{F}_i$ contains the i-th rank individuals.

Then, the individuals in $\mathcal{F}$ move to $\mathcal{P}_{g+1}$ on a rank by rank basis, starting with $\mathcal{F}_1$ (Lines 23 to 26). If the number of individuals in $\mathcal{P}_{g+1} \cup \mathcal{F}_i$ is less than N, $\mathcal{F}_i$ moves to $\mathcal{P}_{g+1}$. Otherwise, a subset of $\mathcal{F}_i$ moves to $\mathcal{P}_{g+1}$. The subset is selected based on the crowding distance metric, which measures the distribution (or diversity) of individuals in the objective space (Deb et al.,

2000) (Lines 27 to 29). The metric computes the distance between two closest neighbors of an individual in each objective and sums up the distances associated with all objectives. A higher crowding distance means that an individual in question is more distant from its neighboring individuals in the objective space. In Line 28, the individuals in $\mathcal{F}_i$ are sorted based on their crowding distance measures, from the one with the highest crowding distance to the one with the lowest crowding distance. The individuals with higher crowding distance measures have higher chances to be selected to $\mathcal{P}_{g+1}$ (Line 29).

```
 1  g = 0
 2  P_g = Randomly generated N individuals
 3  while g < MAX-GENERATION do
 4    O_g = ∅
 5    while |O_g| < N do
 6      p_1 = tournament(P_g)
 7      p_2 = tournament(P_g)
 8      if random() ≤ P_c then
 9        {o_1,o_2} = crossover(p_1, p_2)
10        if random() ≤ P_m then
11          o_1 = mutation(o_1)
12        end if
13        if random() ≤ P_m then
14          o_2 = mutation(o_2)
15        end if
16        O_g = {o_1,o_2} ∪ O_g
17      end if
18    end for
19    R_g = P_g ∪ O_g
20    F = sortByDominationRanking(R_g)
21    P_{g+1} = {∅}
22    i = 1
23    while |P_{g+1}| + |F_i| ≤ N do
24      P_{g+1} = P_{g+1} ∪ F_i
25      i = i + 1
26    end while
27    assignCrowdingDistance(F_i)
28    sortByCrowdingDistance(F_i)
29    P_{g+1} = P_{g+1} ∪ F_i[1 : (N − |P_{g+1}|)]
30    g = g + 1
31  end while
```

Listing 1. Optimization Process in the Proposed EMOA

3.3 Crossover

The proposed EMOA adopts partially-mapped crossover (PMX) as its crossover operator. PMX was originally proposed to solve TSP (Goldbert & Lingle, 1985). It is known that PMX effectively works for TSP and its variants (Goldbert & Lingle, 1985; Kellegőz et al., 2008).

PMX first selects two crossover points on parent individuals at random. A sub-route surrounded by the two crossover points is called a mapping section. In an example in Figure 2, parent 1's mapping section is [3, 9, 4, 13], and parent 2's mapping section is [2, 8, 7, 3]. Given two mapping sections, mapping relationships are formed by paring elements in the mapping sections on a position by position basis. In Figure 2, the first elements in two mapping sections, 2 and 3, are paired; 2–3 is the first mapping relationship. Similarly, three extra mapping relationships, 8–9, 7–4 and 3–13, are formed. In order to reproduce two offspring from two

parents, mapping sections are exchanged between parents. In Figure 2, parent 1's mapping section is replaced with parent 2's; therefore, one of proto-offspring is [0, 2, 7, 6, 2, 8, 7, 3, 10, 5, 1, 0]. (Note that Figure 2 does not show the other proto-offspring.) If proto-offspring has redundant vertices across its mapping section and the other section, PMX replaces each redundant vertex with its counterpart shown in mapping relationships. In Figure 2, 7 and 2 are redundant vertices. Given a mapping relationship of 7–4, 7 is replaced with 4 in the non-mapping section. (Replacements always occur in the non-mapping section.) 2 is replaced with 13 by referencing two mapping relationships (2–3 and 3–13) recursively.

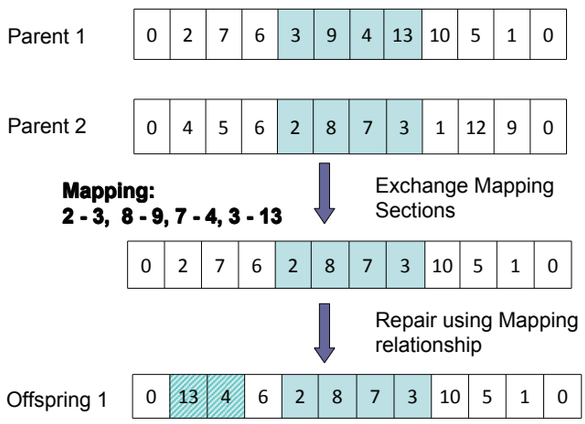


Fig. 2. An Example Crossover (PMX) Process

3.4 Mutation

The proposed EMOA provides a multi-mode mutation operator to alter reproduced offspring. The operator has the following four modes and selects one of them at a time randomly.

1. *Add*: randomly chooses a vertex from unvisited vertices and inserts it to a randomly-selected position in a route (Figure 3(a)). This mode gives the salesman a higher chance to visit more vertices.
2. *Delete*: removes a randomly-selected vertex from a route(Figure 3(b)). This mode reduces the number of vertices that the salesman visits.
3. *Exchange*: randomly chooses a vertex in a route and replaces it with one of unvisited vertices (Figure 3(c)). The unvisited vertex is also selected at random. This mode is intended to change a set of vertices that the salesman visits.
4. *Swap*: exchanges the positions of two randomly-selected nodes in a route (Figure 3(a)). This mode is intended to change a visiting sequence of vertices.

3.5 α-Dominance

This section describes the notion of α-dominance and the design of the α-dominance operator. α-dominance is a new dominance relationship that extends a classical dominance relationship described in Section 1. It takes objective value samples of given two individuals, estimates the impacts of noise on the samples, and determines whether it is confident enough to judge which one is superior/inferior between the two individuals.

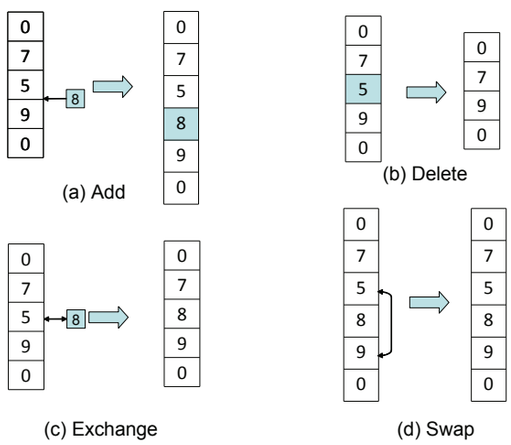


Fig. 3. Example Mutation Processes

3.5.1 The α-dominance operator

The α-dominance operator determines the α-dominance relationship between given two individuals by statistically processing their objective value samples. With this operator, individual A is said to *α-dominate* individual B (denoted by $A \succ_\alpha B$), iif:

- A's and B's objective value samples are classifiable with a statistical confidence level of α, and
- $\mathcal{C}(A,B) = 1 \wedge \mathcal{C}(B,A) < 1$.

In order to examine the first condition, the α-dominance operator classifies A's and B's objective value samples with Support Vector Machine (SVM), and measures a classification error. (See Step 1 in an example shown in Figure 4.) The error (e) is computed as the ratio of the number of miss-classified samples to the total number of samples. For evaluating the confidence level (α) in a classification error, the α-dominance operator computes the classification error's confidence interval (e_{int}):

$$e_{int} = e \pm t_{\alpha,n-1}\sigma \tag{6}$$

$t_{\alpha,n-1}$ denotes a single-tail t-distribution with α confidence level and $n-1$ degrees of freedom. n denotes the total number of samples. σ is the standard deviation of e. It is approximated as follows.

$$\sigma \cong \sqrt{\frac{e}{n}} \tag{7}$$

If e_{int} is significant (i.e., if e_{int} does not span zero), the α-dominance operator cannot classify A's and B's samples with the confidence level of α. Thus, the operator determines that A and B do not α-dominate each other. (See Step 2 in Figure 4.)
If e_{int} is not significant (i.e., if e_{int} spans zero), the α-dominance operator can classify A's and B's samples with the confidence level of α. Thus, the operator examine the aforementioned second condition. (See Step 2 in an example shown in Figure 4.) It measures $\mathcal{C}$-metric (Zitzler & Thiele, 1999) with a classical notion of dominance ($\succ$) described in Section 1.$\mathcal{C}(A,B)$ denotes the fraction of individual B's samples that at least one sample of individual A dominates:

$$\mathcal{C}(A,B) = \frac{|\{b \in B \mid \exists a \in A : a \succ b\}|}{|B|} \tag{8}$$

If $\mathcal{C}(A,B) = 1$, all of B's samples are dominated by at least one sample of A. If $\mathcal{C}(B,A) < 1$, not all of A's samples are dominated by at least one sample of B. The α-dominance operator determines $A \succ_\alpha B$ if $\mathcal{C}(A,B) = 1$ and $\mathcal{C}(B,A) < 1$. If $\mathcal{C}(A,B) < 1$ and $\mathcal{C}(B,A) < 1$, the operator determines neither $A \succ_\alpha B$ nor $B \succ_\alpha A$. See Figure 4 as well.

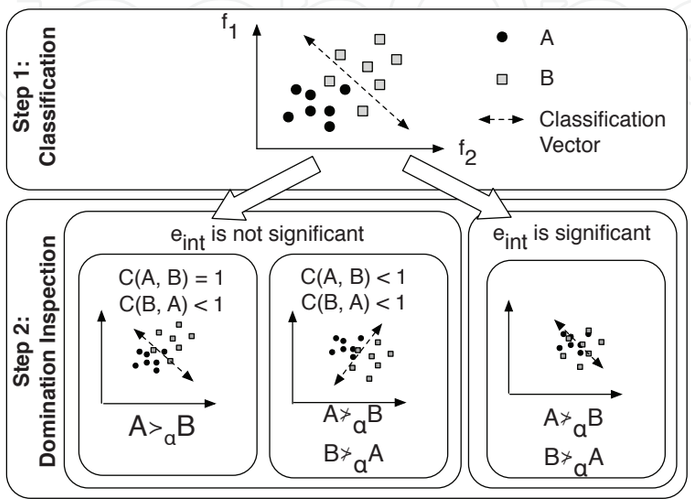


Fig. 4. An Example Process to determine the α-Dominance Relationship between two individuals (A and B)

Figure 4 shows an example to determine the α-dominance relationship between two individuals, A and B, with two objectives, f_1 and f_2, to be minimized. Individual A and B have seven samples each. First, the α-dominance operator classifies these 14 samples in the objective space with SVM and computes e_{int}. Suppose SVM produces a classification vector as shown in Figure 4. Two samples of B are miss-classified; $e = \frac{2}{14}$ (0.143). Thus, $\sigma \cong \sqrt{\frac{0.143}{14}} = 0.1$. Assuming the confidence level α of 95%, $e_{int} = 0.143 \pm 1.771 * 0.1 = 0.143 \pm 0.1771$. Since e_{int} spans zero, A's and B's samples are classifiable with the confidence level of 95%. This means that the aforementioned first condition is hold. In order to examine the second condition, the α-dominance operator measures $\mathcal{C}(A,B)$ and $\mathcal{C}(B,A)$. In Figure 4, $\mathcal{C}(A,B) = 1$ and $\mathcal{C}(B,A) = 2/14 < 1$. This means that the second condition is hold. As a re result, the α-dominance operator concludes $A \succ_\alpha B$.

Listing 2 shows pseudo code of the α-dominance operator. $\mathcal{A}$ and $\mathcal{B}$ denote individual A's and B's samples, respectively. $\mathcal{A}'$ and $\mathcal{B}'$ denote two clusters of samples classified by SVM.

```
1  function alphaDominance(A, B, α)
2    {A', B'} = SVMClassifier(A, B)
3    e = 0 // classification error
4    for each x ∈ A' do
5      if x ∉ A then
6        e = e + 1
7      end if
8    end for
9
10   for each x ∈ B' do
```

```
11      if x ∉ B then
12        e = e + 1
13      end if
14    end for
15
16    if e = 0 then
17      if C(A,B) = 1 then
18        return 1 // A α-dominates B.
19      else if C(B,A) = 1 then
20        return -1 // B α-dominates A.
21      else
22        return 0 // A & B are non-α-dominated.
23      end if
24    else
25      t = t-test(α, sqrt(e,|A| + |B|))
26      if  e − t < 0 then // e_int spans zero.
27        return 0 // A & B are non-α-dominated.
28      else
29        if C(A,B) = 1 then
30          return 1 // A α-dominates B.
31        else if C(B,A) = 1 then
32          return -1 // B α-dominates A.
33        else
34          return 0 // A & B are non-α-dominated.
35        end if
36      end if
37    end if
38  end function
```

Listing 2. Pseudocode of the α-Dominance Operator

3.5.2 Dynamic adjustment of confidence level

The α-dominance operator dynamically adjusts its confidence level (α) by estimating how close individuals have converged to the Pareto-optimal front. The convergence of individuals is estimated based on their disorderliness in the objective space. When individuals are disordered in the objective space, it indicates that they have not converged enough to the Pareto-optimal front. Therefore, the α-dominance operator maintains a low confidence level to determine the α-dominance relationships among individuals in a less strict manner and have diverse individuals explore the decision space and seek the Pareto front. Conversely, when individuals are ordered in the objective space, which indicates that individuals have converged close to the Pareto front, the α-dominance operator increases its confidence level to perform dominance operation in a more strict manner.

The α-dominance operator measures the disorderliness of individuals as their entropy in the objective space. To this end, a hypercube is created in the objective space. Its size is bounded by the maximum and minimum objective values yielded by individuals. (Note that all samples of all individuals, including dominated or non-dominated ones, are plotted in the objective space.) The hypercube is divided to sub-cubes. For example, Figure 5 shows six individuals plotted in a three dimensional hypercube contains eight sub-cubes.

The entropy of individuals (H) is computed as:

$$\left.\begin{array}{r} H = -\sum_{i \in C} P(i) \log_2(P(i)) \\ P(i) = \frac{n_i}{\sum_{i \in C} n_i} \end{array}\right\} \tag{9}$$

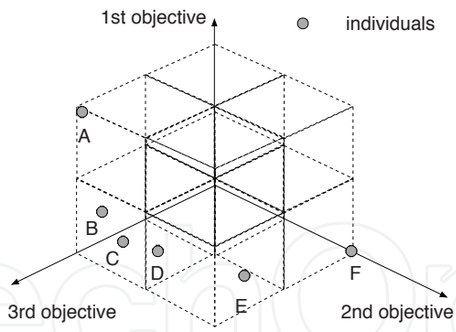


Fig. 5. An Example Hypercube in the Objective Space

C denotes a set of sub-cubes in a hypercube, and $P(i)$ denotes the probability that individuals exist in the i-th sub-cube. n_i denotes the number of individuals in the i-th sub-cube. Entropy (H) is normalized as follows:

$$H_o = \frac{H}{H_{\max}} = \frac{H}{\log_2 n} \tag{10}$$

H_{max} is the maximum entropy: the entropy in the case that all sub-cubes have the same number of individuals. n denotes the total number of sub-cubes. Given normalized entropy (H_o), the confidence level α is adjusted as follows:

$$\alpha = ((\alpha_{\max} - \alpha_{\min})\sqrt{1 - (1 - H_o)^2}) + \alpha_{\min} \tag{11}$$

$\alpha_{\max}$ and $\alpha_{\min}$ denote the predefined maximum and minimum confidence levels, respectively. α is adjusted in a non-linear fashion; a unit circle function is used to map H_o to α.

3.5.3 Integration of the α-dominance operator with parent selection and individual ranking operators

This section describes how α-dominance operator is integrated with the parent selection operator (`tournament()`; Lines 6 and 7 in Listing 1) and the individual ranking operator (
`sortByDominatinoRanking()`; Line 20 in Listing 1).
Listing 3 shows pseudo code of a noise-aware parent selection (binary tournament) operator that leverages the α-dominance operator. $\mathcal{P}$ is the current population of individuals.

```
 1  function tournament(P)
 2    a = randomSelection(P)
 3    b = randomSelection(P)
 4    A = samplesOf(a)
 5    B = samplesOf(b)
 6    r = alphaDominance(A, B, α)
 7    if r = 1 then
 8      return a
 9    else if r = -1 then
10      return b
11    else if r = 0 then
12      if random() > 0.5 then
13        return a
14      else
15        return b
```

```
16      end if
17    end if
18  end function
```

Listing 3. Pseudocode of a Noise-aware Binary Tournament Operator using the α-Dominance Operator

First, two individuals (a and b) are randomly drawn from the current population $\mathcal{P}$ (Lines 2 and 3). Then, in Lines 4 and 5, their samples are obtained to execute the α-dominance operator in Line 6. Depending on the operator's return value (r), one of two individuals (a or b) is returned as a parent individual (Line 7 to 15). If the α-dominance operator cannot determine the α-dominance relationship between a and b (i.e., if $r = 0$), one of them is randomly selected and returned.
Listing 4 shows pseudo code of a noise-aware individual ranking operator that leverages the α-dominance operator. sortByDominatinoRanking() calls `findNonDominatedFront()`, which identifies non-α-dominated individuals in a given population using theα-dominance operator (Lines 11 to 27).

```
1   function sortByDominationRanking(P)
2     i = 1
3     while P ≠ ∅ do
4       F_i = findNonDominatedIndividuals(P)
5       P = P \ F_i
6       i = i + 1
7     end while
8     return F
9   end function
10
11  function findNonDominatedIndividuals(P)
12    P' = ∅
13    for each p ∈ P and p ∉ P' do
14      P' = P' ∪ {p}
15      for each q ∈ P' and q ≠ p do
16      P' = P' ∪ {p}
17      for each q ∈ P' and q ≠ p do
18        A = samplesOf(p)
19        B = samplesOf(q)
20        r = alphaDominance(A, B, α)
21        if r = 1 then
22          P' = P' \ {q}
23        else if r = −1 then
24          P' = P' \ {p}
25        end if
26      end for
27    end for
28    return P'
29  end function
```

Listing 4. Pseudocode of a Noise-aware Individual Ranking Operator using the α-Dominance Operator

4. Experimental evaluation

This section evaluates the proposed EMOA, particularly its α-dominance operator, through a series of computational experiments.

4.1 Test problems

This evaluation study uses three test problems that are built based on three TSP instances: ch130, pr226 and lin318. The TSP instances are obtained from TSPLIB*(Reinelt, 1991). ch130, pr226 and lin318 contain 130, 226 and 318 vertices, respectively. They are customized in this evaluation study so that each vertex maintains a profit and a visiting probability. The value ranges of a profit and a visiting probability are [1.0, 100.0] and [0.0,1.0], respectively. Both values are assigned to each vertex at a uniformly random.

A certain noise is generated and injected to each of two objective functions every time it is evaluated, as shown in Equation 5. Two types of noise are generated: random noise, which follows continuous uniform distributions, and Gaussian noise, which follow normal distributions. Each noise type has three levels of noise: low, medium and high. Table 1 illustrates noise configurations. For random noise, each cell of the table shows a pair of the lower and upper bounds of noise values. For Gaussian noise, each cell of the table shows a pair of the mean and variance of noise values.

		Random noise (Uniform distribution)			Gaussian noise (Normal distribution)		
		ch130	pr226	lin318	ch130	pr226	lin318
Cost	Low	[-20,20]	[-320,320]	[-96,96]	(0,40)	(0,740)	(0,192)
	Medium	[-80,80]	[-1280,1280]	[-384,384]	(0,100)	(0,1600)	(0,480)
	High	[-140,140]	[-2240,2240]	[-672,672]	(0,160)	(0,2560)	(0,768)
Profit	Low	[-2,2]	[-2,2]	[-2,2]	(0,4)	(0,4)	(0,4)
	Medium	[-8,8]	[-8,8]	[-8,8]	(0,10)	(0,10)	(0,10)
	High	[-14,14]	[-14,14]	[-14,14]	(0,16)	(0,16)	(0,16)

Table 1. Noise Configurations for Costs and Profits

4.2 Algorithmic and experimental configurations

The proposed EMOA is configured with a set of parameters shown in Table 2. It is called NSGA-II-A, or simply A, in this evaluation study because it follows NSGA-II's algorithmic structure and customizes the structure with the α-dominance operator, the PMX crossover operator and a mutation operator described in Section 3.4 In order to evaluate the α-dominance operator, NSGA-II-A is compared with the following three variants of NSGA-II:

- NSGA-II (or simply R): the original NSGA-II (Deb et al., 2000) with its crossover and mutation operators replaced by PMX and a mutation operator described in Section 3.4 . Its classical dominance operator does not consider noise in objective functions.
- NSGA-II-U (or simply U): NSGA-II with its classical dominance operator by a noise-aware dominance operator that assumes uniform distribution noise (Teich, 2001).
- NSGA-II-N (or simply N): NSGA-II with its classical dominance operator by a noise-aware dominance operator that assumes normal distribution noise (Eskandari et al., 2007).

All experiments have been implemented and carried out with jMetal (Durillo et al., 2006). Every experimental result is obtained and shown based on 20 independent experiments.

*http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/

Parameter	Value	Parameter	Value
Population size	100	# of samples per individual	30
Max # of generations	500	SVM type	C-support vector classification
Crossover rate	0.9		
Mutation rate	0.2	SVM kernel	Linear
$\alpha_{\min}$ in Equation 11	0.90	C parameter for SVM	1
$\alpha_{\max}$ in Equation 11	0.99	SVM termination criteria	$1e^{-3}$

Table 2. Parameter Configurations

4.3 Metrics for performance evaluation

This evaluation study uses the following five performance metrics to compare individual algorithms.

- *The number of non-dominated individuals*: counts the number of non-dominated individuals in the population at the last (i.e., the 500th) generation. The higher this number is, the more successfully an algorithm in question has evolved and converged individuals by eliminating dominated ones. This metric evaluates the degree of convergence/evolution pressure on individuals.
- *Hypervolume* (Zitzler & Thiele, 1999): measures the volume that non-dominated individuals cover in the objective space. The higher a hypervolume measure is, the closer non-dominated individuals are to the Pareto-optima. This metric evaluates the optimality of individuals.
- $D1_R$ (Knowles & Corne, 2002): is computed as follows.

$$D1_R(A) = \frac{1}{|R|} \sum_{\mathbf{r} \in R} min_{\mathbf{z} \in A} d(\mathbf{r}, \mathbf{z}) \tag{12}$$

  A denotes a set of non-dominated individuals. R denotes a set of reference individuals. $d(\mathbf{r}, \mathbf{z}) = max_k \frac{(r_k - z_k)}{R_k}$ where $k = 1, ..., K$ indexes objectives and R_k is the range of objective values that individuals in R yield with respect to the k-th objective. r_k and z_k denote the objective values that individuals r and z yield with respect to the k-th objective. In this evaluation study, R contains a set of non-dominated individuals that NSGA-II produces when no noise is given to objective functions. Therefore, the lower a $D1_R$ measure is, the more effectively an algorithm in question cancels the existence of noise to yield a more similar performance as NSGA-II's. This metric evaluates the optimality of individuals as well as their degree of noise canceling.
- *U-metric* (Leung & Wang, 2003): is computed as follows.

$$U = \frac{1}{D} \sum_{i=1}^{D} \left| \frac{d_i}{\bar{d}} - 1 \right| \tag{13}$$

  d_i denotes the euclidean distance between the i-th individual and its nearest neighbor in the objective space. D denotes the total number of pairs of the nearest neighbors among non-dominated individuals. $\bar{d} = \frac{1}{D} \sum_{i=1}^{D} d_i$ is the average of d_i. The lower a U-metric measure is, the more uniformly individuals are distributed in the objective space. This metric evaluates the distribution (or diversity) of individuals.

- *C-Metric* (Zitzler & Thiele, 1999): uses Equation 8 to compare two different algorithms by examining the two sets of non-dominated individuals they produce.

In addition to the above metrics, this evaluation study examines the objective values that the non-dominated individuals of each algorithm yield at the last generation.

4.4 The number of non-dominated individuals in the population

Tables 3 and 4 show the number of non-dominated individuals that individual algorithms produces at the last generation. Its average and standard deviation results are obtained based on 20 independent experiments. A bold font face is used to indicate the best result(s) in each noise level case of each problem.

Tables 3 and 4 demonstrate that NSGA-II-A and NSGA-II consistently yield the best and worst results, respectively, in both cases with normal and uniform distribution noises. Under normal distribution noise, NSGA-II-A produces 95 or more non-dominated individuals, while NSGA-II produces only 32.2 in the high noise case of lin318. Under uniform distribution noise, NSGA-II-A evolves all individuals to be non-dominated in all cases, while NSGA-II produces only 49.7 non-dominated individuals in the high noise case of lin318. Tables 3 and 4 illustrate that the α-dominance operator successfully retains a high pressure to evolve and converge

Problem	Noise level	NSGA-II-A		NSGA-II		NSGA-II-N		NSGA-II-U	
		Avg	Sd	Avg	Sd	Avg	Sd	Avg	Sd
ch130	Low	**100**	0	74.7	9.89	**100**	0	**100**	0
	Medium	**99.2**	2.8	52.5	9.1	98.8	2.7	93.2	3.9
	High	**98.9**	0	34.1	9.3	90.1	8.9	89.5	2.1
pr226	Low	**100**	0	68.2	6.9	**100**	0	94.2	5.1
	Medium	**99.0**	12.5	42.3	4.7	93.2	8.2	86.5	7.9
	High	**97.1**	5.87	30.1	7.6	89.5	5.9	80.4	13.5
lin318	Low	**100**	0	66.3	9.5	99.5	7.8	96.5	3.5
	Medium	**99.5**	1.8	49.35	8.6	90.4	5.2	83.8	16.5
	High	**95.0**	12.5	32.2	7.2	85.3	4.6	80.1	12.4

Table 3. The Number of Non-dominated Individuals at the last Generation when Normal Distribution Noise is injected to Objective Functions

Problem	Noise level	NSGA-II-A		NSGA-II		NSGA-II-N		NSGA-II-U	
		Avg	Sd	Avg	Sd	Avg	Sd	Avg	Sd
ch130	Low	**100**	0	96.9	5.7	**100**	0	**100**	0
	Medium	**100**	0	56.2	10.4	98.1	2.1	**100**	0
	High	**100**	0	43.3	9.9	92.3	4.5	98.4	2.1
pr226	Low	**100**	0	83.8	11.1	**100**	0	100	0
	Medium	**100**	0	56.5	9.9	86.5	5.4	**100**	0
	High	**100**	0	46.3	7.9	80.6	10.2	90.0	4.8
lin318	Low	**100**	0	73.4	9.1	**100**	0	**100**	0
	Medium	**100**	0	58.4	6.9	89.9	8.7	98.4	1.1
	High	**100**	0	49.7	8.7	69.6	11.6	89.3	8.1

Table 4. The Number of Non-dominated Individuals at the last Generation when Uniform Distribution Noise is injected to Objective Functions

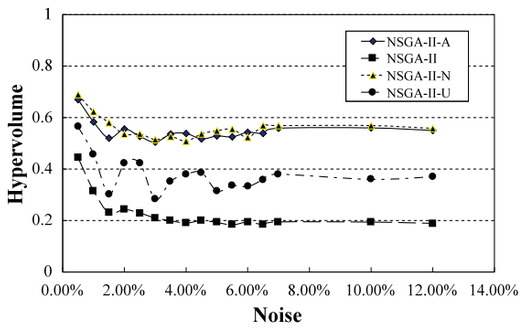


Fig. 6. Hypervolume (ch130, Normal Distribution Noise)

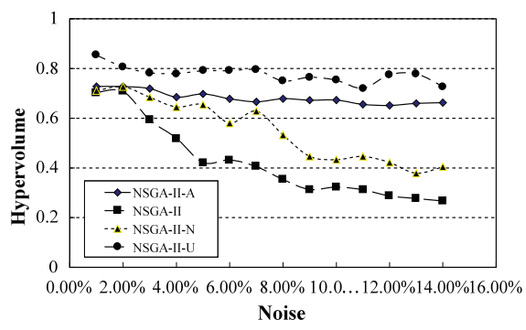


Fig. 7. Hypervolume (ch130, Uniform Distribution Noise)

individuals even in a harder problem (lin318) with a high level of noise. On the contrary, NSGA-II significantly loses the pressure as a given problem becomes harder and a given noise level becomes higher because its dominance operator does not handle noise at all.

In Table 3, NSGA-II-N produces 100 non-dominated individuals in two cases and yields better results than NSGA-II and NSGA-II-U because it assumes normal distribution noise beforehand. However, NSGA-II-A outperforms NSGA-II-N as a given problem becomes harder and a given noise level becomes higher. A similar observation is made in Figure 4; NSGA-II-A outperforms NSGA-II-U even when uniform distribution noise is injected to objective functions, as a given problem becomes harder and a given noise level becomes higher.

4.5 Optimality evaluation with hypervolume

Figures 6 to 11 show the hypervolume measures that individual algorithms yield in each problem with different noise distributions. NSGA-II-A yields the best hypervolume results in most cases (except in ch130 with uniform distribution; Figure 7). NSGA-II yields the worst results in most cases (except in pr226 with normal distribution; Figure 9), which is reasonable because its dominance operator does not handle noise in objective functions.

In Figure 6 (ch130 with normal distribution noise), NSGA-II-A performs similarly to NSGA-II-N, which outperforms NSGA-II and NSGA-II-U, because it designed to handle normal distribution noise. As a given problem becomes harder with normal distribution noise, NSGA-II-A yields better hypervolume measures than NSGA-II-N. (See Figures 8 and 10.) A similar observation is made in Figures 7, 9 and 11. In Figure 7, NSGA-II-A is outperformed by NSGA-II-U, which is designed to handle uniform distribution noise. However, it outperforms NSGA-II-U in harder problems (Figures 9 and 11). These results demonstrate that the α-dominance operator allows NSGA-II-A to successfully seek quality solutions toward the Pareto-optima regardless of problems and noise distributions.

In general, all algorithms yield smaller hypervolume measures as the amount of noise increases. However, the hypervolume measures of NSGA-II-A do not vary largely under different noise levels. It can maintain the hypervolume of 0.6 or higher in all the cases. NSGA-II, NSGA-II-N (under uniform distribution noise) and NSGA-II-U (under normal distribution noise) significantly decrease their hypervolume measures as given noise levels increases. In the pr226 problem with the highest normal distribution noise, NSGA-II-A performs 32.8% better than NSGA-II-U, 50.8% better than NSGA-II and 67.2% better than

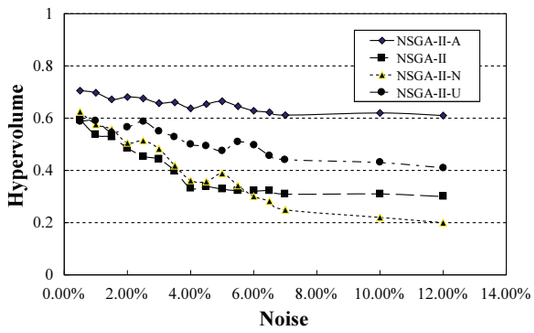


Fig. 8. Hypervolume (pr226, Normal Distribution Noise)

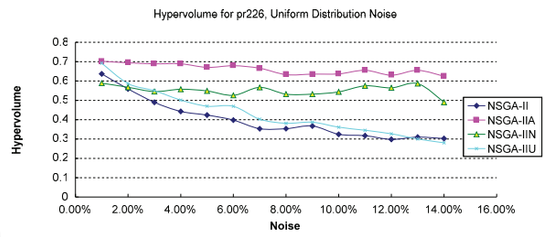


Fig. 9. Hypervolume (pr226, Uniform Distribution Noise)

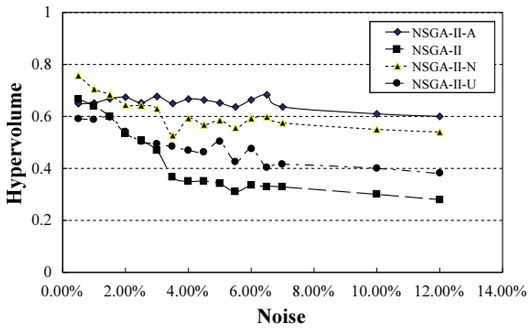


Fig. 10. Hypervolume (lin318, Normal Distribution Noise)

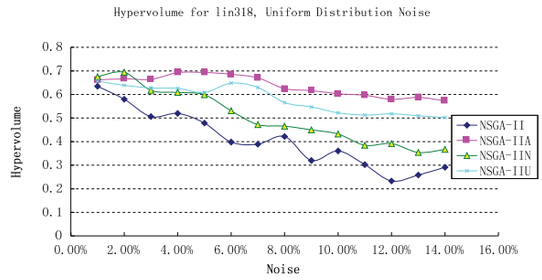


Fig. 11. Hypervolume (lin318, Uniform Distribution Noise)

NSGA-II-N. These results demonstrate that the α-dominance operator is less sensitive against noise levels than existing noise-aware dominance operators in NSGA-II-U and NSGA-II-N.

4.6 Evaluation of optimality and noise canceling with $D1_R$

Figures 12 to 17 show the $D1_R$ measures that individual algorithms yield in each problem with different noise distributions. As Figures 12, 14 and 16 depict, when normal distribution noise is injected to objective functions, NSGA-II-U performs poorly. This is understandable because it does not expect normal distribution noise at all. The other three algorithms perform similarly. Although NSGA-II-A is outperformed by NSGA-II-N and NSGA-II in the cs130 problem (Figure 12), it outperforms them in the other harder problems (the pr226 and lin318 problems; Figures 14 and 16). In the lin318 problem, which is hardest in the three test problems in this evaluation study, NSGA-II-A exhibits its superiority over NSGA-II-N and other algorithms.
When uniform distribution noise is injected to objective functions, NSGA-II-A outperforms the other three algorithms in all problems, although all algorithms perform similarly, particularly in the ch130 problem (Figures 13, 15 and 17). NSGA-II-A exhibits its superiority in the lin318 problem than the other two problems. These results illustrates that the α-dominance operator allows NSGA-II-A to successfully suppress the impacts of noise on the evolution/convergence of individuals and seek quality solutions toward the Pareto-optima.
As Figures 12 to 17 show, $D1_R$ measures grow in all algorithms when the amount of injected noise increases. This means that the evolution/convergence of individuals suffers a higher

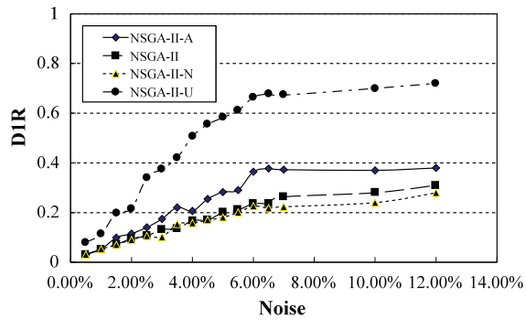


Fig. 12. $D1_R$ (ch130, Normal Dist. Noise)

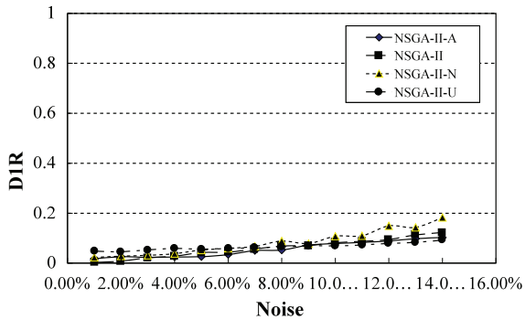


Fig. 13. $D1_R$ (ch130, Uniform Dist. Noise)

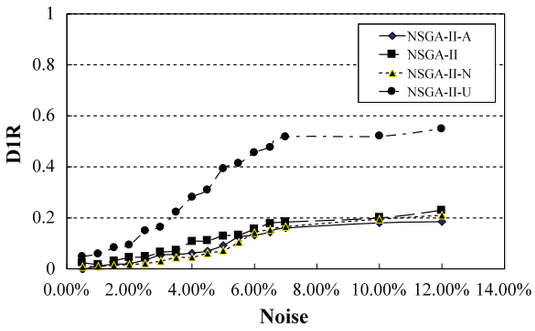


Fig. 14. $D1_R$ (pr226, Normal Dist. Noise)

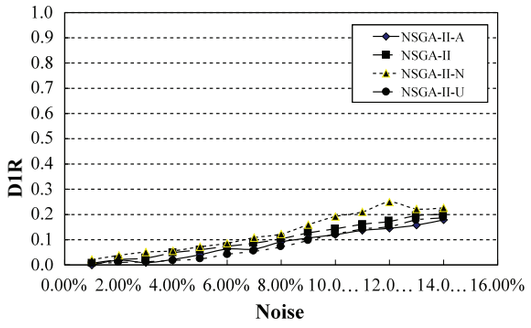


Fig. 15. $D1_R$ (pr226, Uniform Dist. Noise)

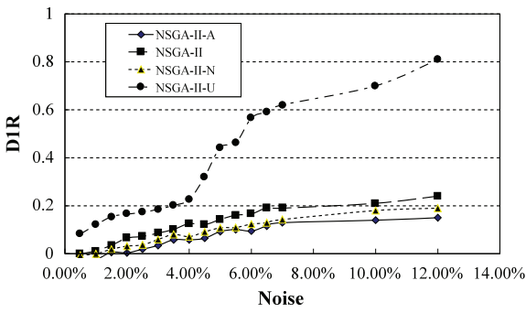


Fig. 16. $D1_R$ (lin318, Normal Dist. Noise)

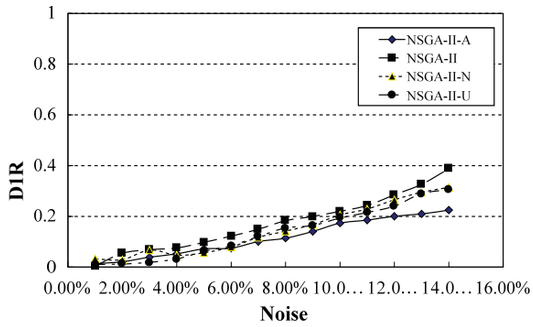


Fig. 17. $D1_R$ (lin318, Uniform Dist. Noise)

interference by a higher noise level. This trend is consistent with the results of hypervolume measures (Section 4.5). However, the $D1_R$ measures of NSGA-II-A do not vary largely under different noise levels. NSGA-II-A can maintain the $D1_R$ measure of approximately 0.2 or lower in all the cases except the ch130 problem with normal distribution noise (Figures12). This contrasts with, for example, NSGA-II-U that significantly increases $D1_R$ under higher noise levels in all problems. In the lin 318 problem with normal distribution noise, NSGA-II-A's $D1_R$ is under 0.2 under the highest noise level while NSGA-II-U's $D1_R$ exceeds 0.8. These results demonstrate that the α-dominance operator is less sensitive against noise levels than existing noise-aware dominance operators in NSGA-II-U and NSGA-II-N.

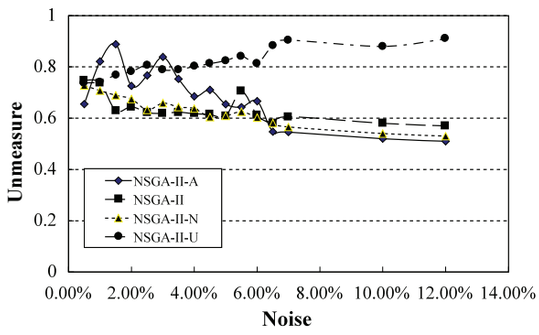


Fig. 18. U-metric (ch130, Normal Dist. Noise)

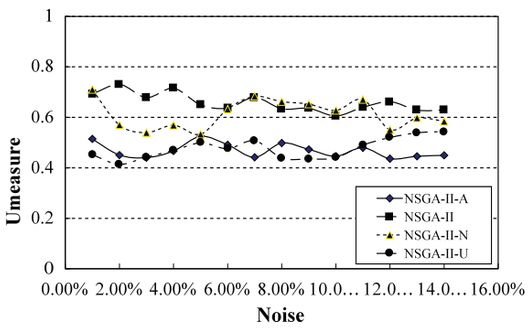


Fig. 19. U-metric (ch130, Uniform Dist. Noise)

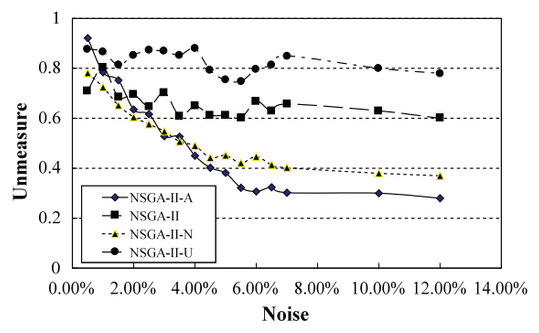


Fig. 20. U-metric (pr226, Normal Dist. Noise)

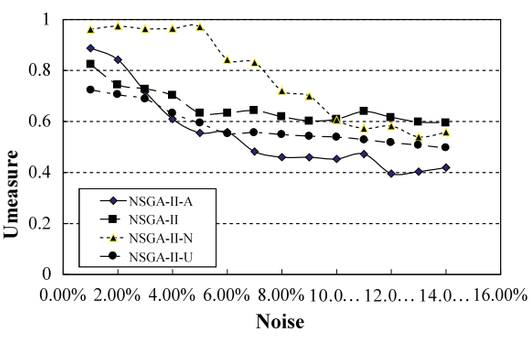


Fig. 21. U-metric (pr226, Uniform Dist. Noise)

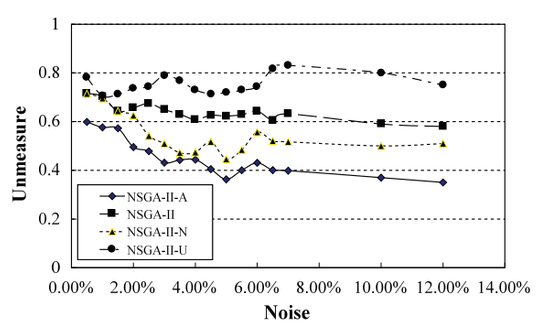


Fig. 22. U-metric (lin318, Normal Dist. Noise)

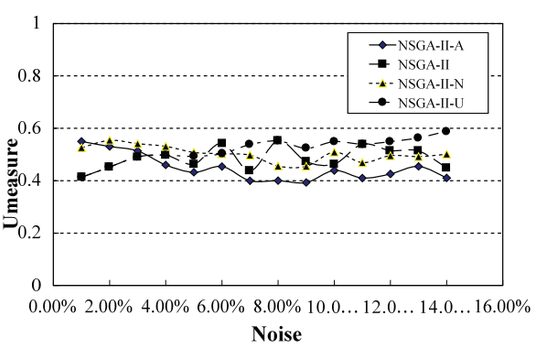


Fig. 23. U-metric (lin318, Uniform Dist. Noise)

4.7 Diversity evaluation with U-metric

Figures 18 to 23 show the U-metric measures that individual algorithms yield in each problem with different noise distributions. The four algorithms perform more similarly to each other in the problems with uniform distribution noise than normal distribution noise. In all cases, NSGA-II-A yields the best U-metric measures under the highest noise level. Considering that all algorithms perform the same (NSGA-II's) crowding distance operator for diversity preservation among individuals, the α-dominance operator produces the lowest interfere to diversity preservation based on crowding distance.

	Noise Level	$\mathcal{C}$(NSGA-II-A, NSGA-II)	$\mathcal{C}$(NSGA-II, NSGA-II-A)	$\mathcal{C}$(NSGA-II-A, NSGA-II-U)	$\mathcal{C}$(NSGA-II-U, NSGA-II-A)	$\mathcal{C}$(NSGA-II-A, NSGA-II-N)	$\mathcal{C}$(NSGA-II-N, NSGA-II-A)
ch130	Low	**4.23e − 03**	1.05e-03	0.00e+00	0.00e+00	0.00e+00	0.00e+00
	Medium	**2.28e − 03**	0.00e+00	**3.21e − 03**	1.25e-03	0.00e+00	0.00e+00
	High	**1.20e − 03**	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
pr226	Low	**5.01e − 03**	0.00e+00	**5.23e − 03**	1.11e-03	0.00e+00	0.00e+00
	Medium	**0.21e − 03**	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
	High	**3.10e − 03**	1.01e-03	0.00e+00	0.00e+00	0.00e+00	0.00e+00
lin318	Low	**8.27e − 03**	1.10e-03	**2.01e − 03**	0.00e+00	0.00e+00	0.00e+00
	Medium	**2.22e − 03**	0.00e+00	**1.00e − 03**	0.00e+00	0.00e+00	0.00e+00
	High	**9.36e − 03**	2.01e-03	0.00e+00	0.00e+00	0.00e+00	0.00e+00

Table 5. $\mathcal{C}$-metric Measures under Normal Distribution Noise

4.8 Algorithm comparison with C-metric

Tables 5 and 6 show the $\mathcal{C}$-metric measures to compare individual algorithms with each other under normal distribution noise and uniform distribution noise, respectively.

With normal distribution noise injected to objective functions (Tables 5), $\mathcal{C}$(NSGA-II-A, NSGA-II) > $\mathcal{C}$(NSGA-II, NSGA-II-A) in all of nine cases. (A bold font face is used to indicate a higher $\mathcal{C}$-metric measure between $\mathcal{C}$(NSGA-II-A, NSGA-II) and $\mathcal{C}$(NSGA-II, NSGA-II-A).) This means that NSGA-II-A outperforms NSGA-II in all the cases. In five of nine cases, NSGA-II produces no individuals that dominate the ones produced by NSGA-II-A. In comparison between NSGA-II-A and NSGA-II-U, $\mathcal{C}$(NSGA-II-A, NSGA-II-U) > $\mathcal{C}$(NSGA-II-U, NSGA-II-A) in three of nine cases. This means that NSGA-II-A outperforms NSGA-II-U in the three cases and the two algorithms tie in the other six cases. In seven of nine cases, NSGA-II-U produces no individuals that dominate the ones produced by NSGA-II-A. In comparison between NSGA-II-A and NSGA-II-N, the two algorithm tie in all nine cases. Even though NSGA-II-N is designed to handle normal distribution noise, it produces no individuals that dominate the ones produced by NSGA-II-A. Note that NSGA-II-A often outperforms NSGA-II-N in harder problems with higher normal distribution noise in terms of the number of non-dominated individuals (Section 4.4), hypervolume (Section 4.5) and $D1_R$ (Section 4.6).

	Noise Level	$\mathcal{C}$(NSGA-II-A, NSGA-II)	$\mathcal{C}$(NSGA-II, NSGA-II-A)	$\mathcal{C}$(NSGA-II-A, NSGA-II-U)	$\mathcal{C}$(NSGA-II-U, NSGA-II-A)	$\mathcal{C}$(NSGA-II-A, NSGA-II-N)	$\mathcal{C}$(NSGA-II-N, NSGA-II-A)
ch130	Low	0.00e+00	0.00e+00	0.00e+00	0.00e+00	1.01e-03	**2.01e − 03**
	Medium	**1.11e − 03**	0.00e+00	**4.01e − 03**	0.00e+00	**2.10e − 03**	1.30e-03
	High	0.00e+00	0.00e+00	0.00e+00	0.00e+00	**2.58e − 03**	0.00e+00
pr226	Low	**1.22e − 03**	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
	Medium	**4.29e − 03**	0.00e+00	**1.01e − 03**	0.00e+00	**6.07e − 03**	3.12e-03
	High	**5.18e − 03**	2.14e-03	**0.89e − 03**	0.00e+00	**1.00e − 03**	0.00e+00
lin318	Low	0.00e+00	0.00e+00	0.00e+00	0.00e+00	**2.01e − 03**	0.00e+00
	Medium	**12.28e − 03**	0.00e+00	0.00e+00	0.00e+00	**6.31e − 03**	0.00e+00
	High	**2.33e − 03**	0.00e+00	**5.14e − 03**	1.02e-03	**1.08e − 03**	0.00e+00

Table 6. $\mathcal{C}$-Metric Measures under Uniform Distribution Noise

With uniform distribution noise injected to objective functions (Tables 6), NSGA-II-A outperforms NSGA-II in six of nine cases. In eight cases, NSGA-II produces no individuals that dominate the ones produced by NSGA-II-A. In comparison between NSGA-II-A and NSGA-II-U, which assumes uniform distribution noise in advance, NSGA-II-A outperforms NSGA-II-U in four of nine cases, and the two algorithms tie in the other five cases. In eight

cases, NSGA-II-U produces no individuals that dominate the ones produced by NSGA-II-A. A general observation is that NSGA-II-A outperforms NSGA-II-U in harder problems with higher uniform distribution noise. It is consistent with the observations made with the number of non-dominated individuals (Section 4.4), hypervolume (Section 4.5) and $D1_R$ (Section 4.6). In comparison between NSGA-II-A and NSGA-II-N, NSGA-II-A outperforms NSGA-II-N in seven of nine cases. In six cases, NSGA-II-N produces no individuals that dominate the ones produced by NSGA-II-A.

As Tables 5 and 6 illustrate, NSGA-II-N performs better under normal distribution noise than uniform distribution noise. This is reasonable because it anticipates normal distribution noise in advance. However, it never outperforms NSGA-II-A in Table- 5). On the contrary, NSGA-II-U performs poorly under both normal and uniform distribution noise. It never outperforms NSGA-II-A in Tables 5 and 6. These results demonstrate that, although the α-dominance operator assumes no noise distribution in advance, it performs under both normal and uniform distribution noise. Moreover, it exhibits higher superiority under higher noise levels.

4.9 Optimality evaluation with objective values

Tables 7 and 8 show the objective values that each algorithm's individuals yield at the last (the 500th) generation under normal and uniform distribution noise, respectively. L, M and H mean that low, medium and high levels of noise. Each table shows the average and standard deviation results that are obtained from 20 independent experiments. A bold font face is used to indicate the best average result among four algorithms on an objective by objective basis. An asterisk (*) is placed for an average result when the result is significantly different (worse) than the best average result based on a t-test with 19 degrees of freedom and 95% confidence level.

Table 7 depicts that, in the ch130 problem, NSGA-II-A is the best among four algorithms in cost under the low noise level, in profit under the medium noise level, and in both objectives under the high noise level. Under the high noise level, NSGA-II-A exhibits statistical significance over NSGA-II and NSGA-II-N in cost and over NSGA-II and NSGA-II-U in profit. In the pr220 problem, NSGA-II-A is the best among four algorithms in cost under all noise levels. It is also the best in profit under the low and high levels of noise. Under the high noise level, NSGA-II-A exhibits statistical significance over NSGA-II-N and NSGA-II-U in cost and over NSGA-II-R and NSGA-II-U in profit. Moreover, NSGA-II-A yields the lowest standard deviation in both objectives in all noise levels. This means that the variance of it's objective values is the lowest among different experiments. In the lin318 problem, NSGA-II-A yields the best objective values and the best standard deviation values in both objectives under all noise levels. Under the high noise level, it exhibits statistical significance over NSGA-II and NSGA-II-U in cost and over all the other three algorithms in profit. Table 7 demonstrates that the α-dominance operator allows NSGA-II-A to outperform the other algorithms more often in more metrics as a given problem becomes harder and a given noise level becomes higher.

Table 8 shows qualitatively similar results to the ones in Table 7. In the lin318 problem with the highest noise level, NSGA-II-A yields the best objective values and the best standard deviation values in both objectives under all noise levels. It exhibits statistical significance over NSGA-II and NSGA-II-N in cost and over all the other three algorithms in profit.

		ch130				pr220				lin318			
		Cost		profit		Cost		profit		Cost		profit	
		Avg	Sd	Avg	Sd	Avg	Sd	Avg	Sd	Avg	Sd	Avg	Sd
L	A	**2533.8**	174.6	3129.0⋆	34.2	**92236.2**	181.2	**6828.3**	82.4	**45353.2**	204.3	**8852.7**	111.0
	R	2890.5⋆	167.9	3124.3	46.4	116225.3⋆	192.5	6826.9	95.3	47927.6	248.2	8748.9	118.3
	N	2646.2⋆	186.7	**3208.8**	49.4	104295.2⋆	188.7	6806.4	107.4	48918.3⋆	212.2	8398.4⋆	152.2
	U	2760.1⋆	214.3	2958.6⋆	39.2	114219.1⋆	214.0	6485.0⋆	111.0	50688.4⋆	302.3	8539.0⋆	185.8
M	A	3471.6	244.8	**3855.1**	79.2	**132637.8**	242.4	7077.6	93.1	**45894.5**	402.3	**8740.5**	185.4
	R	**3280.9**	272.2	3768.2	76.4	136017.3	290.4	**7153.1**	142.6	56162.1⋆	499.8	8520.8⋆	321.0
	N	3971.8⋆	266.1	3685.7⋆	76.9	135070.4	345.7	6944.9⋆	112.6	56649.0⋆	461.0	8522.2⋆	201.8
	U	3698.3⋆	291.9	3602.3⋆	85.8	139797.3⋆	293.2	6920.4⋆	150.2	61204.3⋆	481.4	8622.2	342.4
H	A	**4564.2**	312.5	**3734.6**	84.1	**148905.8**	243.2	**6752.2**	101.0	**56227.3**	527.4	**8504.8**	385.1
	R	4740.3⋆	248.5	3534.6⋆	93.1	150162.1	277.8	6557.8⋆	161.4	70204.6⋆	599.4	8361.9⋆	414.4
	N	4702.5⋆	324.2	3656.2	89.4	171468.0⋆	267.3	6733.6	143.3	56450.7	582.2	8292.8⋆	442.1
	U	4615.1	352.8	3593.6⋆	98.7	171936.6⋆	391.0	6436.6⋆	165.0	66661.1⋆	701.3	8258.3⋆	499.5

Table 7. Objective Values at the Last Generation under Normal Distribution Noise

		ch130				pr220				lin318			
		Cost		profit		Cost		profit		Cost		profit	
		Avg	Sd	Avg	Sd	Avg	Sd	Avg	Sd	Avg	Sd	Avg	Sd
L	A	**2368.5**	242.4	3139.9	183.1	**99782.9**	144.2	**6949.3**	131.0	**41937.1**	309.7	**9010.6**	119.9
	R	2777.2⋆	223.6	3167.6	184.1	105385.1⋆	214.0	6566.0⋆	152.1	43865.8	351.9	9000.9	134.7
	N	2466.0	265.9	2895.2⋆	100.3	122022.7⋆	183.5	6779.7⋆	141.1	53194.4⋆	408.6	8937.6⋆	177.4
	U	2860.1⋆	281.2	**3252.0**	78.6	110096.8⋆	188.3	6920.4	158.8	50819.3⋆	469.5	8805.6⋆	102.4
M	A	**2976.3**	203.2	3117.2	146.3	**97879.5**	248.3	**7137.8**	136.8	**48109.6**	677.9	**8829.7**	87.7
	R	3705.6⋆	296.2	2666.7⋆	145.6	110639.5⋆	321.0	6925.2⋆	148.5	53918.6⋆	686.5	8374.6⋆	123.6
	N	3573.3⋆	210.3	2709.3⋆	163.5	119108.1⋆	295.4	6680.0⋆	146.1	53346.2⋆	707.5	8442.0⋆	118.4
	U	3368.7	189.3	**3344.5**	135.3	122815.9⋆	274.1	6816.6⋆	148.5	53640.5⋆	706.9	8519.1⋆	171.0
H	A	4384.6	404.1	**3838.0**	137.7	**151578.5**	381.1	**7022.4**	132.2	**61174.5**	439.8	**8887.9**	114.8
	R	4463.5⋆	426.7	2798.0⋆	142.6	164282.2⋆	458.4	6523.2⋆	139.8	66035.3⋆	647.3	8152.2⋆	267.4
	N	4992.5⋆	445.8	2730.0⋆	152.8	178198.2⋆	422.2	6830.6⋆	136.9	62668.5⋆	553.8	8566.8⋆	168.0
	U	**4299.3**	499.1	3330.0⋆	163.6	166907.1⋆	461.9	6711.4⋆	142.5	61701.4	520.3	8285.4⋆	189.1

Table 8. Objective Values at the Last Generation under Uniform Distribution Noise

5. Related work

pTSP and TSPP have been studied extensively and used to model many real-world applications in different fields (Feillet et al., 2005). Early pTSP studies adopted heuristics that were modified from the heuristics to solve TSP (e.g., nearest neighbor, savings heuristic, k-opt exchanges, 1-shift) (Bertsimas, 1988; Birattari et al., 2007). Recent studies often focus on meta-heuristics, such as ant colony optimization algorithms (Branke & Guntsch, 2004) and evolutionary algorithms (Liu, 2008; Liu et al., 2007), in favor of their global search capabilities. However, these algorithms are not applicable for pTSPP because pTSP is a single objective optimization problem and pTSPP is a multiobjective optimization problem as described in Section 2.

TSPP is a multiobjective optimization algorithm; however, a number of existing work have attempted to solve it as a single objective optimization problem by aggregating multiple objectives into a single fitness function as, for example, a weighted sum of objective values or considering extra objectives as constraints with given bounds (Awerbuch et al., 1999; Laporte & Martello, 1990). These algorithms are not designed to seek the optimal tradeoff (i.e., Pareto-optimal) solutions among conflicting objectives. Moreover, it is not always straightforward to manually tune weight values in a fitness function that aggregates multiple objective values.

A very limited number of existing work have attempted to solve TSPP with multiobjective optimization algorithms (Jozefowiez et al., 2008b). These algorithms better address the characteristics of pTSPP; however, they never consider noise in objective functions.

The α-dominance operator is designed to aid seeking the Pareto-optimality of solution candidates in multiobjective optimization problems with noisy objective functions. In the area of evolutionary multiobjective optimization, there exist several existing work to handle uncertainties in objective functions by modifying NSGA-II's classical dominance operator

(Beyer, 2000; Jin & Branke, 2005). All of them assume particular noise distributions in advance. For example,Babbar et al. (2003); Eskandari et al. (2007); Goh & Tan (2006) assume normal distribution noise. Teich (2001) assume uniform distribution noise. Delibrasis et al. (1996); Wormington et al. (1999) assume Poisson distribution noise. Given a noise distribution, each of existing noise-aware dominance operators statistically estimates each individual's objective value by collecting its samples. In contrast, the α-dominance operator assumes no noise distributions a priori because, in general, it is hard to predict and model them in most (particularly, real-world) multiobjective optimization problems. Instead of estimating each individual's objective values, the α-dominance operator estimates the impacts of noise on objective value samples and determines whether it is confident enough to compare individuals.

Another line of relevant research is to handle uncertainties in decision variables (Deb & Gupta, 2006; Deb et al., 2009; 2006). These work proposes the notion of robust individuals, and the robustness quantifies the sensitivity of noise in the decision space on the objective space. They also assume normal distribution noise in advance. Unlike these work, α-dominance focuses on uncertainties in the objective space and assumes no noise distributions in advance.

6. Conclusions

This chapter formulates a noisy multiobjective optimization problem, the Probabilistic Traveling Salesman Problem with Profits (pTSPP), which contains noise in its objective functions. In order to solve pTSPP, this chapter proposes an evolutionary multiobjective optimization algorithm (EMOA) that leverages a novel noise-aware dominance operator, called the α-dominance operator. The operator takes objective value samples of given two individuals, estimates the impacts of noise on the samples and determines whether it is statistically confident enough to judge which individual is superior/inferior to the other. Experimental results demonstrate that the α-dominance operator allows the proposed EMOA to effectively obtain quality solutions to pTSPP and it outperforms existing noise-aware dominance operators.

7. References

Arnold, D. (2000). Evolution strategies in noisy environments – a survey of existing work, *in* L. Kallel, B. Naudts & A. Rogers (eds), *Theoretical Aspects of Evolutionary Computing*, Springer, chapter 11.

Awerbuch, B., Azar, Y., Blum, A. & Vempala, S. (1999). New approximation guarantees for minimum-weight k-trees and prize-collecting salesmen, *SIAM J. Comput.* 28(1): 254–262.

Babbar, M., Lakshmikantha, A. & Goldberg, D. E. (2003). A modified NSGA-II to solve noisy multiobjective problems, *Proc. of ACM Genetic and Evol. Comp. Conf.*

Bertsimas, D. (1988). Probabilistic combinatorial optimization problems, *Ph.D dissertation, MIT, USA* .

Bertsimas, D. & Howell, L. H. (1993). Further results on the probabilistic traveling salesman problem, *European Journal of Operational Research* 65(1): 68–95.

Beyer, H.-G. (2000). Evolutionary algorithms in noisy environments: Theoretical issues and guidelines for practice, *Elsevier Computer Methods in Applied Mechanics and Engineering* 186(2–4).

Beyer, H.-G. & Sendhoff, B. (2007). Robust Optimization – A Comprehensive Survey, *Computer Methods in Applied Mechanics and Engineering* 196(33-34): 3190–3218.

Bianchi, L., Dorigo, M., Gambardella, L. & Gutjahr, W. (2009). A Survey on Metaheuristics for Stochastic Combinatorial Optimization, *Natural Computing* 8(2): 239–287.

Birattari, M., Balaprakash, P., Stï£¡ï£¡tzle, T. & Dorigo, M. (2007). Estimation-based local search for the probabilistic traveling salesman problem, *MIC07: The seventh Metaheuristics International Conference* .

Branke, J. & Guntsch, M. (2004). Solving the probabilistic tsp with ant colony optimization, *Journal of Mathematical Modelling and Algorithms* 3: 403–425.

Carroll, R. J., Ruppert, D., Stefanski, L. A. & Crainiceanu, C. (2006). *Measurement Error in Nonlinear Models: A Modern Perspective, Second Edition*, CRC Press.

Deb, K., Agrawal, S., Pratab, A. & Meyarivan, T. (2000). A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II, *Proc. of Int'l Parallel Problem Solving from Nature Conf.*

Deb, K. & Gupta, H. (2006). Introducing robustness in multi-objective optimization, *Evol. Comput.* 14: 463–494.

Deb, K., Gupta, S., Daum, D., Branke, J., Mall, A. K. & Padmanabhan, D. (2009). Reliability-based optimization using evolutionary algorithms, *Trans. Evol. Comp* 13: 1054–1074.

Deb, K., Padmanabhan, D., Gupta, S. & Mall, A. K. (2006). Handling uncertainties through reliability-based optimization using evolutionary algorithms, *Technical Report KanGAL 2006009*, Indian Institute of Technology Kanpur.

Delibrasis, K., Undrill, P. & Cameron, G. (1996). Genetic algorithm implementation of stack filter design for image restoration, *Proc. of Vision, image and signal proc.*

Diwekar, U. & Kalagnanam, J. (1997). Efficient Sampling Technique for Optimization under Uncertainty, *Wiley AIChE J.* 43(2).

Durillo, J. J., Nebro, A. J., Luna, F., Dorronsoro, B. & Alba, E. (2006). jMetal: A java framework for developing multi-objective optimization metaheuristics, *Technical Report ITI-2006-10*, Departamento de Lenguajes y Ciencias de la Computación, University of Málaga.

Eskandari, H., Geiger, C. D. & Bird, R. (2007). Handling uncertainty in evolutionary multiobjective optimization: SPGA, *Proc. of IEEE Congress on Evol. Comp.*

Feillet, D., Dejax, P. & Gendreau, M. (2005). Traveling salesman problems with profits, *INFORMS Transportation Science* 39(2): 188–205.

Goh, C. K. & Tan, K. C. (2006). Noise handling in evolutionary multi-objective optimization, *Proc. of IEEE Congress on Evol. Comp.*

Goldbert, D. E. & Lingle, R. (1985). Alleles, loci, and the traveling salesman problem, *Proc. of the Second Int. Conf. on Genetic Algorithms and Their Applications*, pp. 154–159.

Jaillet, P. (1985). *Probabilistic Traveling Salesman Problem*, PhD thesis, Massachusetts Institution of Technology.

Jin, Y. & Branke, J. (2005). Evolutionary optimization in uncertain environments-a survey, *IEEE Trans. on Evol. Comp.* 9.

Jozefowiez, N., Glover, F. & Laguna, M. (2008a). Multi-objective meta-heuristics for the traveling salesman problem with profits, *Springer Journal of Mathematical Modelling and Algorithms* 7(2): 177–195.

Jozefowiez, N., Glover, F. & Laguna, M. (2008b). Multi-objective meta-heuristics for the traveling salesman problem with profits, *Journal of Mathematical Modelling and Algorithms* 7: 177–195.

Kellegőz, T., Toklu, B. & Wilson, J. (2008). Comparing efficiencies of genetic crossover operators for one machine total weighted tardiness problem, *Applied Mathematics and Computation* 199(2): 590 – 598.

Knowles, J. & Corne, D. (2002). On metrics for comparing nondominated sets, *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*, Vol. 1, pp. 711 –716.

Laporte, G. & Martello, S. (1990). The selective travelling salesman problem, *Discrete Appl. Math.* 26(2-3): 193–207.

Leung, Y.-W. & Wang, Y.-P. (2003). U-measure: A quality measure for multiobjective programming, *Technical Report COMP-03-011*, Hong kong Baptist University.

Liu, Y.-H. (2008). *Solving the Probabilistic Traveling Salesman Problem Based on Genetic Algorithm with Queen Selection Scheme*, InTech, chapter Traveling Salesman Problem.

Liu, Y.-H., Jou, R.-C., Wang, C.-C. & Chiu, C.-S. (2007). An evolutionary algorithm with diversified crossover operator for the heterogeneous probabilistic tsp, *MDAI '07: Proceedings of the 4th international conference on Modeling Decisions for Artificial Intelligence*, pp. 351–360.

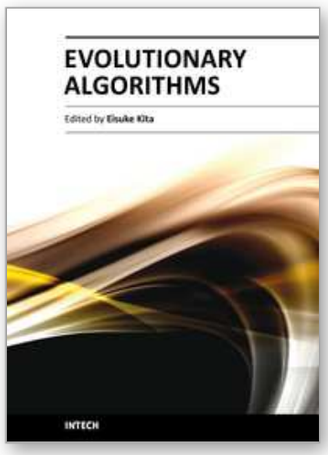
Reinelt, G. (1991). Tsplib – a traveling salesman problem library, *ORSA Journal on Computing* 3: 376.

Srinivas, N. & Deb, K. (1995). Multiobjective function optimization using nondominated sorting genetic algorithms, *MIT Evol. Comp. J.* 2(3).

Teich, J. (2001). Pareto-front exploration with uncertain objectives, *Proc. of Int'l Conf. on Evol. Multi-Criterion Optimization*.

Wormington, M., Panaccione, C., Matney, K. M. & Bowen, D. K. (1999). Characterization of structures from x-ray scattering data using genetic algorithms, *JSTOR Philosophical Transactions* 357(1761).

Zitzler, E. & Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach, *IEEE Trans. Evol. Comp.* 3(4).



Evolutionary Algorithms

Edited by Prof. Eisuke Kita

ISBN 978-953-307-171-8
Hard cover, 584 pages
Publisher InTech
Published online 26, April, 2011
Published in print edition April, 2011

Evolutionary algorithms are successively applied to wide optimization problems in the engineering, marketing, operations research, and social science, such as include scheduling, genetics, material selection, structural design and so on. Apart from mathematical optimization problems, evolutionary algorithms have also been used as an experimental framework within biological evolution and natural selection in the field of artificial life.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen