The Tunable W-Model Benchmark Problem

Thomas Weise & Zijun Wu · 汤卫思 & 吴自军

tweise@hfuu.edu.cn · http://iao.hfuu.edu.cn

Hefei University, South Campus 2
Faculty of Computer Science and Technology
Institute of Applied Optimization
230601 Shushan District, Hefei, Anhui, China
Econ. & Tech. Devel. Zone, Jinxiu Dadao 99

合肥学院 南艳湖校区/南2区
计算机科学与技术系
应用优化研究所
中国 安徽省 合肥市 蜀山区 230601
经济技术开发区 锦绣大道99号

July 24, 2018

Outline

1. Introduction

2. The *W-Model*

3. Experiment

4. Summary

website

Section Outline

These are the slides for paper [1] presented at the BB-DOB workshop at GECCO'2018.

1. Introduction
2. The *W-Model*
3. Experiment
4. Summary

website

What do we want from a Benchmark Problem?

What do we want from a Benchmark Problem?

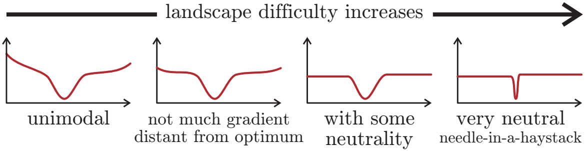
- The goal of benchmarking is to get a complete picture of the strengths and weaknesses of optimization methods.

What do we want from a Benchmark Problem?

- The goal of benchmarking is to get a complete picture of the strengths and weaknesses of optimization methods.
- Some frequently occurring problem characteristics cause difficulties to optimization algorithms [2, 3]

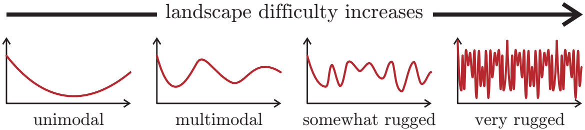
What do we want from a Benchmark Problem?

- The goal of benchmarking is to get a complete picture of the strengths and weaknesses of optimization methods.
- Some frequently occurring problem characteristics cause difficulties to optimization algorithms [2, 3]
  - neutrality



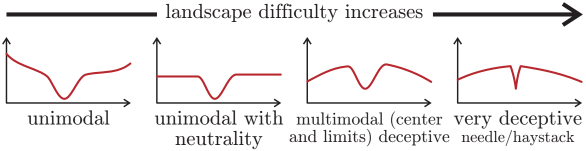
What do we want from a Benchmark Problem?

- The goal of benchmarking is to get a complete picture of the strengths and weaknesses of optimization methods.
- Some frequently occurring problem characteristics cause difficulties to optimization algorithms [2, 3]
    - neutrality
    - ruggedness



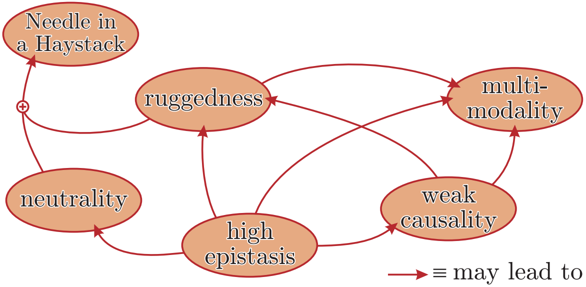
What do we want from a Benchmark Problem?

- The goal of benchmarking is to get a complete picture of the strengths and weaknesses of optimization methods.
- Some frequently occurring problem characteristics cause difficulties to optimization algorithms [2, 3]
  - neutrality
  - ruggedness
  - deceptiveness


landscape difficulty increases
unimodal
unimodal with neutrality
multimodal (center and limits) deceptive
very deceptive
needle/haystack

What do we want from a Benchmark Problem?

- The goal of benchmarking is to get a complete picture of the strengths and weaknesses of optimization methods.
- Some frequently occurring problem characteristics cause difficulties to optimization algorithms [2, 3]
  - neutrality
  - ruggedness
  - deceptiveness
  - epistasis



A Benchmark Problem should. . .

A Benchmark Problem should. . .

1. include problems which exhibit the difficult features ruggedness, epistasis, neutrality, and deceptiveness in different strengths and in different combinations.

A Benchmark Problem should...

1. include problems which exhibit the difficult features ruggedness, epistasis, neutrality, and deceptiveness in different strengths and in different combinations.

- So are classical problems like the Traveling Salesman Problem (TSP) [4, 5] or the Maximum Satisfiability Problem (SAT) [6, 7] good candidates?

A Benchmark Problem should...

1. include problems which exhibit the difficult features ruggedness, epistasis, neutrality, and deceptiveness in different strengths and in different combinations.

- So are classical problems like the Traveling Salesman Problem (TSP) [4, 5] or the Maximum Satisfiability Problem (SAT) [6, 7] good candidates?
- Not really. It is not that easy to understand how hard, difficult, rugged, deceptive, or epistatic a TSP or SAT problem is...

A Benchmark Problem should. . .

1. include problems which exhibit the difficult features ruggedness, epistasis, neutrality, and deceptiveness in different strengths and in different combinations.
2. exhibit these difficult features in degrees which are obvious, easy to understand, and ideally tunable.

A Benchmark Problem should. . .

1. include problems which exhibit the difficult features ruggedness, epistasis, neutrality, and deceptiveness in different strengths and in different combinations.
2. exhibit these difficult features in degrees which are obvious, easy to understand, and ideally tunable.
3. have a problem hardness determined directly by tunable parameters.

A Benchmark Problem should...

1. include problems which exhibit the difficult features ruggedness, epistasis, neutrality, and deceptiveness in different strengths and in different combinations.
2. exhibit these difficult features in degrees which are obvious, easy to understand, and ideally tunable.
3. have a problem hardness determined directly by tunable parameters.
4. have known optima and the range of the objective function should be known.

A Benchmark Problem should...

1. include problems which exhibit the difficult features ruggedness, epistasis, neutrality, and deceptiveness in different strengths and in different combinations.
2. exhibit these difficult features in degrees which are obvious, easy to understand, and ideally tunable.
3. have a problem hardness determined directly by tunable parameters.
4. have known optima and the range of the objective function should be known.
5. be easy to understand and fast to compute.

A Benchmark Problem should. . .

1. include problems which exhibit the difficult features ruggedness, epistasis, neutrality, and deceptiveness in different strengths and in different combinations.
2. exhibit these difficult features in degrees which are obvious, easy to understand, and ideally tunable.
3. have a problem hardness determined directly by tunable parameters.
4. have known optima and the range of the objective function should be known.
5. be easy to understand and fast to compute.
6. fit to a standard representation from discrete optimization, i.e., either bit strings or permutations (of fixed length).

A Benchmark Problem should. . .

1. include problems which exhibit the difficult features ruggedness, epistasis, neutrality, and deceptiveness in different strengths and in different combinations.
2. exhibit these difficult features in degrees which are obvious, easy to understand, and ideally tunable.
3. have a problem hardness determined directly by tunable parameters.
4. have known optima and the range of the objective function should be known.
5. be easy to understand and fast to compute.
6. fit to a standard representation from discrete optimization, i.e., either bit strings or permutations (of fixed length).
7. allow the creation of easy and hard, small and large instances.

A Benchmark Problem should. . .

1. include problems which exhibit the difficult features ruggedness, epistasis, neutrality, and deceptiveness in different strengths and in different combinations.
2. exhibit these difficult features in degrees which are obvious, easy to understand, and ideally tunable.
3. have a problem hardness determined directly by tunable parameters.
4. have known optima and the range of the objective function should be known.
5. be easy to understand and fast to compute.
6. fit to a standard representation from discrete optimization, i.e., either bit strings or permutations (of fixed length).
7. allow the creation of easy and hard, small and large instances.
8. be replicable, i.e., allow to derive problem instances deterministically from very few parameters.

A Benchmark Problem should. . .

1. include problems which exhibit the difficult features ruggedness, epistasis, neutrality, and deceptiveness in different strengths and in different combinations.
2. exhibit these difficult features in degrees which are obvious, easy to understand, and ideally tunable.
3. have a problem hardness determined directly by tunable parameters.
4. have known optima and the range of the objective function should be known.
5. be easy to understand and fast to compute.
6. fit to a standard representation from discrete optimization, i.e., either bit strings or permutations (of fixed length).
7. allow the creation of easy and hard, small and large instances.
8. be replicable, i.e., allow to derive problem instances deterministically from very few parameters.
9. be theoretically tractable.

A Benchmark Problem should...

1. include problems which exhibit the difficult features ruggedness, epistasis, neutrality, and deceptiveness in different strengths and in different combinations.
2. exhibit these difficult features in degrees which are obvious, easy to understand, and ideally tunable.
3. have a problem hardness determined directly by tunable parameters.
4. have known optima and the range of the objective function should be known.
5. be easy to understand and fast to compute.
6. fit to a standard representation from discrete optimization, i.e., either bit strings or permutations (of fixed length).
7. allow the creation of easy and hard, small and large instances.
8. be replicable, i.e., allow to derive problem instances deterministically from very few parameters.
9. be theoretically tractable.
10. have components which can be combined with other, existing problems.

A Benchmark Problem should. . .

1. include problems which exhibit the difficult features ruggedness, epistasis, neutrality, and deceptiveness in different strengths and in different combinations.
2. exhibit these difficult features in degrees which are obvious, easy to understand, and ideally tunable.
3. have a problem hardness determined directly by tunable parameters.
4. have known optima and the range of the objective function should be known.
5. be easy to understand and fast to compute.
6. fit to a standard representation from discrete optimization, i.e., either bit strings or permutations (of fixed length).
7. allow the creation of easy and hard, small and large instances.
8. be replicable, i.e., allow to derive problem instances deterministically from very few parameters.
9. be theoretically tractable.
10. have components which can be combined with other, existing problems.
11. be extensible to other domains, e.g., variable-length representations, multi-objective domains, . . .

A Benchmark Problem should...

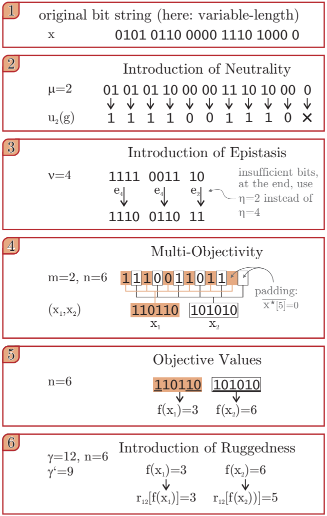
1. include problems which exhibit the difficult features ruggedness, epistasis, neutrality, and deceptiveness in different strengths and in different combinations.
2. exhibit these difficult features in degrees which are obvious, easy to understand, and ideally tunable.
3. have a problem hardness determined directly by tunable parameters.
4. have known optima and the range of the objective function should be known.
5. be easy to understand and fast to compute.
6. fit to a standard representation from discrete optimization, i.e., either bit strings or permutations (of fixed length).
7. allow the creation of easy and hard, small and large instances.
8. be replicable, i.e., allow to derive problem instances deterministically from very few parameters.
9. be theoretically tractable.
10. have components which can be combined with other, existing problems.
11. be extensible to other domains, e.g., variable-length representations, multi-objective domains, ...
12. have an available reference implementation with utilities and tests showing that/whether the implementation (or any implementation of the problem) is identical to the problem definition in a publication, maybe even an experiment execution environment.

A Benchmark Problem should...

1. include problems which exhibit the difficult features ruggedness, epistasis, neutrality, and deceptiveness in different strengths and in different combinations.
2. exhibit these difficult features in degrees which are obvious, easy to understand, and ideally tunable.
3. have a problem hardness determined directly by tunable parameters.
4. have known optima and the range of the objective function should be known.
5. be easy to understand and fast to compute.
6. fit to a standard representation from discrete optimization, i.e., either bit strings or permutations (of fixed length).
7. allow the creation of easy and hard, small and large instances.
8. be replicable, i.e., allow to derive problem instances deterministically from very few parameters.
9. be theoretically tractable.
10. have components which can be combined with other, existing problems.
11. be extensible to other domains, e.g., variable-length representations, multi-objective domains, . . .
12. have an available reference implementation with utilities and tests showing that/whether the implementation (or any implementation of the problem) is identical to the problem definition in a publication, maybe even an experiment execution environment.
13. have available example data sets with results from example experiments.

The W-Model Tunable Benchmark Problem

- Benchmark Model [1, 8, 9] defined over $\{0,1\}^n$.



1 original bit string (here: variable-length)

x 0101 0110 0000 1110 1000 0

2 Introduction of Neutrality

μ=2 01 01 01 10 00 00 11 10 10 00 0

$u_2(g)$ 1 1 1 1 0 0 1 1 1 0 ×

3 Introduction of Epistasis

ν=4 1111 0011 10

e_4 e_4 e_2

1110 0110 11

insufficient bits, at the end, use η=2 instead of η=4

4 Multi-Objectivity

m=2, n=6 1 1 1 0 0 1 1 0 1 1

padding: $\overline{x^\star[5]}=0$

(x_1,x_2) 110110 101010

x_1 x_2

5 Objective Values

n=6 110110 101010

$f(x_1)=3$ $f(x_2)=6$

6 Introduction of Ruggedness

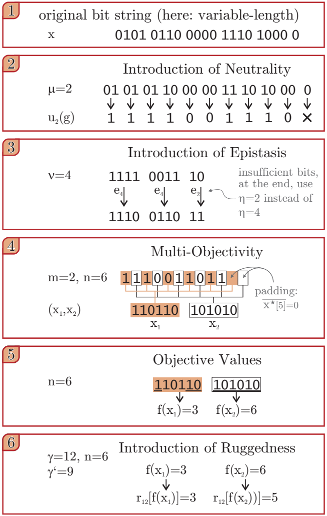
γ=12, n=6

γ‘=9

$f(x_1)=3$ $f(x_2)=6$

$r_{12}[f(x_1)]=3$ $r_{12}[f(x_2))]=5$

The W-Model Tunable Benchmark Problem



1 original bit string (here: variable-length)

x 0101 0110 0000 1110 1000 0

2 Introduction of Neutrality

$\mu=2$ 01 01 01 10 00 00 11 10 10 00 0

$u_2(g)$ 1 1 1 1 0 0 1 1 1 0 ×

3 Introduction of Epistasis

$\nu=4$ 1111 0011 10

e_4 e_4 e_2

1110 0110 11

insufficient bits, at the end, use $\eta=2$ instead of $\eta=4$

4 Multi-Objectivity

m=2, n=6 1 1 1 0 0 1 1 0 1 1

(x_1,x_2) 110110 101010

x_1 x_2

padding: $x^\star[5]=0$

5 Objective Values

n=6 110110 101010

$f(x_1)=3$ $f(x_2)=6$

6 Introduction of Ruggedness

$\gamma=12$, n=6

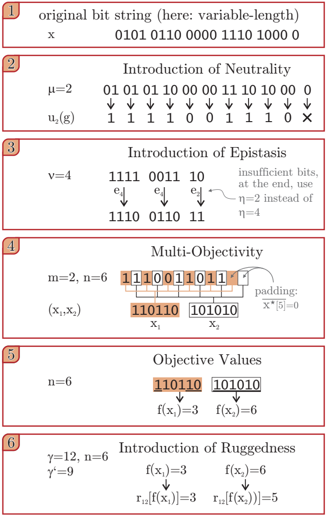
$\gamma'=9$

$f(x_1)=3$ $f(x_2)=6$

$r_{12}[f(x_1)]=3$ $r_{12}[f(x_2))]=5$

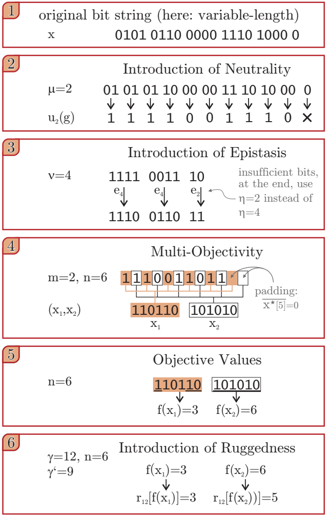
- Benchmark Model [1, 8, 9] defined over $\{0,1\}^n$.
- Fulfills all 13 requirements above.

The W-Model Tunable Benchmark Problem


1
original bit string (here: variable-length)
x 0101 0110 0000 1110 1000 0
2
Introduction of Neutrality
μ=2 01 01 01 10 00 00 11 10 10 00 0
u₂(g) 1 1 1 1 0 0 1 1 1 0 ×
3
Introduction of Epistasis
ν=4 1111 0011 10
e₄ e₄ e₂
1110 0110 11
insufficient bits, at the end, use η=2 instead of η=4
4
Multi-Objectivity
m=2, n=6 1 1 1 0 0 1 1 0 1 1
padding: x*[5]=0
(x₁,x₂) 110110 101010
x₁ x₂
5
Objective Values
n=6 110110 101010
f(x₁)=3 f(x₂)=6
6
Introduction of Ruggedness
γ=12, n=6
γ‘=9
f(x₁)=3 f(x₂)=6
r₁₂[f(x₁)]=3 r₁₂[f(x₂))]=5

- Benchmark Model [1, 8, 9] defined over $\{0,1\}^n$.
- Fulfills all 13 requirements above.
- Neutrality, Epistasis, Ruggedness/Deceptiveness (and multi-objectivity) implemented as separate, parameterized layers which could also be plugged on top of other problems.

The W-Model Tunable Benchmark Problem



1 original bit string (here: variable-length)

x 0101 0110 0000 1110 1000 0

2 Introduction of Neutrality

$\mu=2$ 01 01 01 10 00 00 11 10 10 00 0

$u_2(g)$ 1 1 1 1 0 0 1 1 1 0 ×

3 Introduction of Epistasis

$\nu=4$ 1111 0011 10

e_4 e_4 e_2

1110 0110 11

insufficient bits, at the end, use $\eta=2$ instead of $\eta=4$

4 Multi-Objectivity

m=2, n=6 1 1 1 0 0 1 1 0 1 1

(x_1,x_2) 110110 101010

x_1 x_2

padding: $\overline{x^\star[5]}=0$

5 Objective Values

n=6 110110 101010

$f(x_1)=3$ $f(x_2)=6$

6 Introduction of Ruggedness

$\gamma=12$, n=6

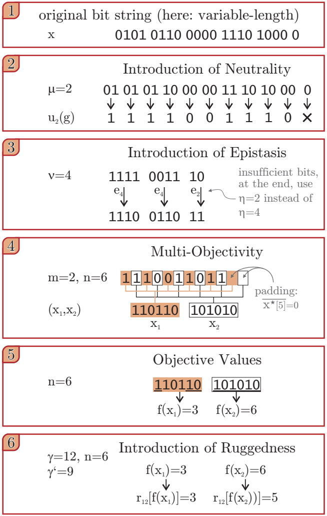
$\gamma'=9$

$f(x_1)=3$ $f(x_2)=6$

$r_{12}[f(x_1)]=3$ $r_{12}[f(x_2))]=5$

- Benchmark Model [1, 8, 9] defined over $\{0,1\}^n$.
- Fulfills all 13 requirements above.
- Neutrality, Epistasis, Ruggedness/Deceptiveness (and multi-objectivity) implemented as separate, parameterized layers which could also be plugged on top of other problems.
- Problem instance completely defined by five parameters n, μ, ν, γ, and m

The W-Model Tunable Benchmark Problem



1 original bit string (here: variable-length)

x 0101 0110 0000 1110 1000 0

2 Introduction of Neutrality

μ=2 01 01 01 10 00 00 11 10 10 00 0

u_2(g) 1 1 1 1 0 0 1 1 1 0 ×

3 Introduction of Epistasis

ν=4 1111 0011 10

e_4 e_4 e_2

1110 0110 11

insufficient bits, at the end, use η=2 instead of η=4

4 Multi-Objectivity

m=2, n=6 1 1 1 0 0 1 1 0 1 1

padding: $x^\star[5]=0$

(x_1,x_2) 110110 101010

x_1 x_2

5 Objective Values

n=6 110110 101010

f(x_1)=3 f(x_2)=6

6 Introduction of Ruggedness

γ=12, n=6

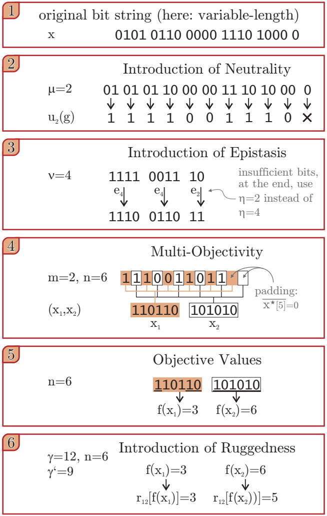
γ'=9

f(x_1)=3 f(x_2)=6

r_{12}[f(x_1)]=3 r_{12}[f(x_2))]=5

- Benchmark Model [1, 8, 9] defined over $\{0,1\}^n$.
- Fulfills all 13 requirements above.
- Neutrality, Epistasis, Ruggedness/Deceptiveness (and multi-objectivity) implemented as separate, parameterized layers which could also be plugged on top of other problems.
- Problem instance completely defined by five parameters n, μ, ν, γ, and m
- Known global optimum: $x^\star =$ 010101010101010...01 of length n with objective value $f(x^\star) = 0$.

The W-Model Tunable Benchmark Problem



1 original bit string (here: variable-length)

x 0101 0110 0000 1110 1000 0

2 Introduction of Neutrality

μ=2 01 01 01 10 00 00 11 10 10 00 0

$u_2(g)$ 1 1 1 1 0 0 1 1 1 0 ✕

3 Introduction of Epistasis

ν=4 1111 0011 10

e_4 e_4 e_2

1110 0110 11

insufficient bits, at the end, use η=2 instead of η=4

4 Multi-Objectivity

m=2, n=6 1 1 1 0 0 1 1 0 1 1

padding: $x^\star[5]=0$

(x_1,x_2) 110110 101010

x_1 x_2

5 Objective Values

n=6 110110 101010

$f(x_1)=3$ $f(x_2)=6$

6 Introduction of Ruggedness

γ=12, n=6

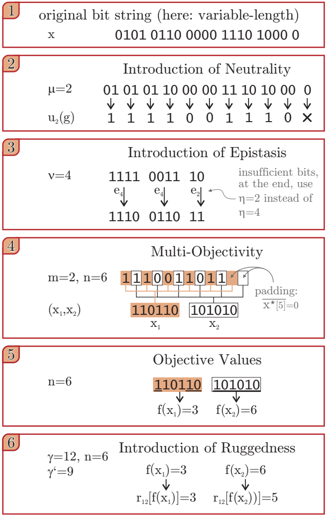
γ‘=9

$f(x_1)=3$ $f(x_2)=6$

$r_{12}[f(x_1)]=3$ $r_{12}[f(x_2))]=5$

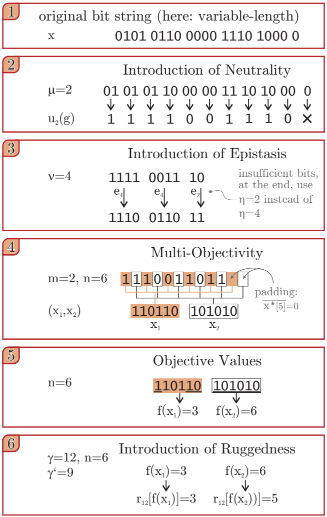
- Benchmark Model [1, 8, 9] defined over $\{0,1\}^n$.
- Fulfills all 13 requirements above.
- Neutrality, Epistasis, Ruggedness/Deceptiveness (and multi-objectivity) implemented as separate, parameterized layers which could also be plugged on top of other problems.
- Problem instance completely defined by five parameters n, μ, ν, γ, and m
- Known global optimum: $x^\star = 010101010101010\ldots01$ of length n with objective value $f(x^\star)=0$.
- Computing of objective function f is in $\mathcal{O}\left(m*n*\nu^2\right)$.

The W-Model Tunable Benchmark Problem



- Benchmark Model [1, 8, 9] defined over $\{0,1\}^n$.
- Fulfills all 13 requirements above.
- …
- Problem instance completely defined by five parameters n, μ, ν, γ, and m
- Known global optimum: $x^\star = 010101010101\ldots01$ of length n with objective value $f(x^\star) = 0$.
- Computing of objective function f is in $\mathcal{O}\left(m * n * \nu^2\right)$.
- Reference implementation [10] with many unit tests, parallel experiment execution environment, and example algorithms at `http://github.com/thomasWeise/BBDOB_W_Model`

The W-Model Tunable Benchmark Problem



1 original bit string (here: variable-length)

x 0101 0110 0000 1110 1000 0

2 Introduction of Neutrality

$\mu=2$ 01 01 01 10 00 00 11 10 10 00 0

$u_2(g)$ 1 1 1 1 0 0 1 1 1 0 ×

3 Introduction of Epistasis

$\nu=4$ 1111 0011 10

e_4 e_4 e_2

1110 0110 11

insufficient bits, at the end, use $\eta=2$ instead of $\eta=4$

4 Multi-Objectivity

m=2, n=6 1 1 1 0 0 1 1 0 1 1 * *

(x_1,x_2) 110110 101010

x_1 x_2

padding: $x^\star[5]=0$

5 Objective Values

n=6 110110 101010

$f(x_1)=3$ $f(x_2)=6$

6 Introduction of Ruggedness

$\gamma=12$, n=6

$\gamma'=9$

$f(x_1)=3$ $f(x_2)=6$

$r_{12}[f(x_1)]=3$ $r_{12}[f(x_2))]=5$

- Benchmark Model [1, 8, 9] defined over $\{0,1\}^n$.
- Fulfills all 13 requirements above.
- . . .
- Computing of objective function f is in $\mathcal{O}\left(m * n * \nu^2\right)$.
- Reference implementation [10] with many unit tests, parallel experiment execution environment, and example algorithms at `http://github.com/thomasWeise/BBDOB_W_Model`
- Huge example experiment [10] obtained from this implementation with 45 GB of algorithm traces at doi:`10.5281/zenodo.1256883`.

Section Outline

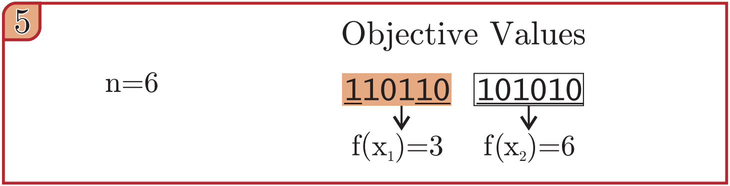
These are the slides for paper [1] presented at the BB-DOB workshop at GECCO'2018.

1. Introduction
2. The *W-Model*
3. Experiment
4. Summary

website

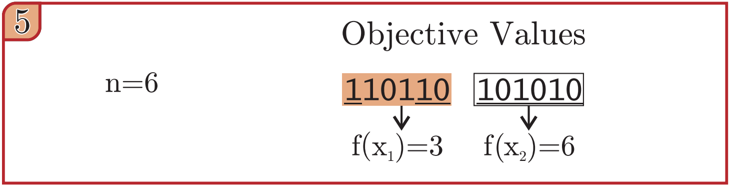
Basic Problem

- Goal: minimize the Hamming distance $h(x, x^{\star})$ to $x^{\star} = (\texttt{0101}\dots)$ of length n.



Basic Problem

- Goal: minimize the Hamming distance $h(x, x^{\star})$ to $x^{\star} = (\texttt{0101}\ldots)$ of length n.
- Very similar to OneMax problem [11–14].



5

Objective Values

n=6

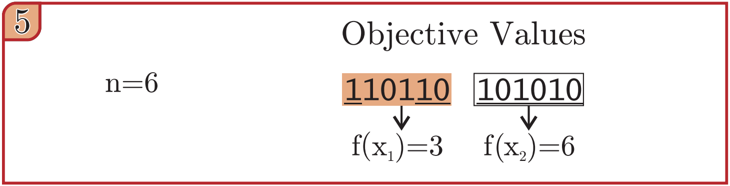
$\underline{1}101\underline{10}$ $\boxed{\underline{101010}}$

$\downarrow$ $\downarrow$

$f(x_1)=3$ $f(x_2)=6$

Basic Problem

- Goal: minimize the Hamming distance $h(x, x^{\star})$ to $x^{\star} = (\texttt{0101}\ldots)$ of length n.
- Very similar to OneMax problem [11–14].
- Computing of objective function is in $\mathcal{O}(n)$ for bit strings with length $l(x) = n$.



5

Objective Values

n=6

110110

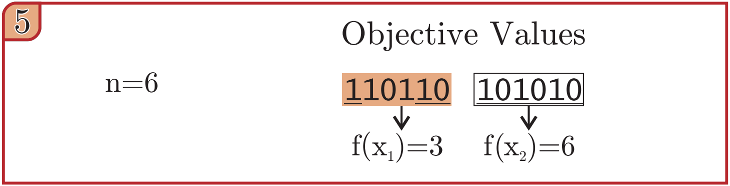
101010

f(x$_1$)=3

f(x$_2$)=6

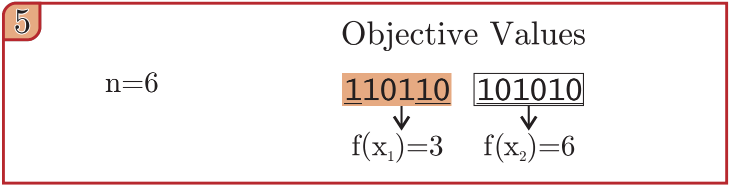
Basic Problem

- Goal: minimize the Hamming distance $h(x, x^\star)$ to $x^\star = (\texttt{0101}\ldots)$ of length n.
- Very similar to OneMax problem [11–14].
- Computing of objective function is in $\mathcal{O}(n)$ for bit strings with length $l(x) = n$.
- Fixed-Length Search Space: bit strings of length n.



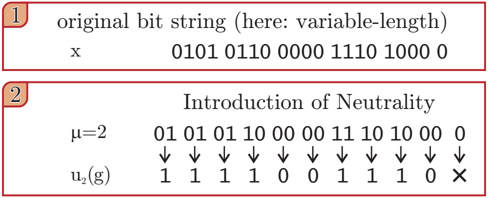
Basic Problem

- Goal: minimize the Hamming distance $h(x, x^{\star})$ to $x^{\star} = (0101\ldots)$ of length n.
- Very similar to OneMax problem [11–14].
- Computing of objective function is in $\mathcal{O}(n)$ for bit strings with length $l(x) = n$.
- Fixed-Length Search Space: bit strings of length n.
- Variable-Length Search Space: overly long strings ($l(x) > n$) are cut after position n, strings that are too short are padded with $\overline{x^{\star}}$.



Neutrality Layer

- The application of a search operator is *neutral* if it yields no change in objective value [15, 16].



1 original bit string (here: variable-length)

x 0101 0110 0000 1110 1000 0

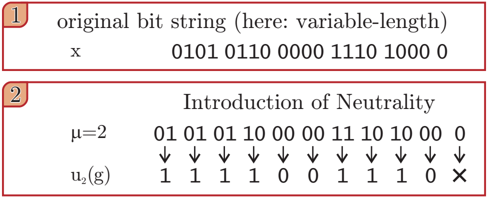
2 Introduction of Neutrality

μ=2 01 01 01 10 00 00 11 10 10 00 0

$u_2(g)$ 1 1 1 1 0 0 1 1 1 0 ✕

Neutrality Layer

- The application of a search operator is *neutral* if it yields no change in objective value [15, 16].
- Usually negative impact on performance [17, 18].



1 original bit string (here: variable-length)

x 0101 0110 0000 1110 1000 0

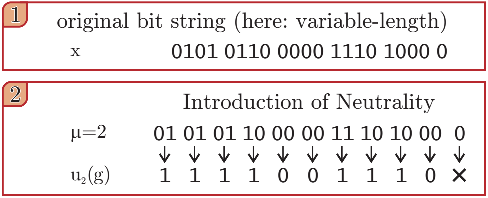
2 Introduction of Neutrality

μ=2 01 01 01 10 00 00 11 10 10 00 0

$u_2(g)$ 1 1 1 1 0 0 1 1 1 0 ✕

Neutrality Layer

- The application of a search operator is *neutral* if it yields no change in objective value [15, 16].
- Usually negative impact on performance [17, 18].
- Transformation $u(x)$ shortens x by factor μ by computing majority value for blocks of length μ, set bit to 1 in draws for even μ (no average effect as $x^{\star} = \texttt{0101}\ldots$).



1

original bit string (here: variable-length)

x 0101 0110 0000 1110 1000 0

2

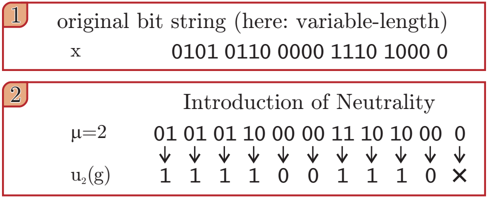
Introduction of Neutrality

μ=2 01 01 01 10 00 00 11 10 10 00 0

u_2(g) 1 1 1 1 0 0 1 1 1 0 ✕

Neutrality Layer

- The application of a search operator is *neutral* if it yields no change in objective value [15, 16].
- Usually negative impact on performance [17, 18].
- Transformation $u(x)$ shortens x by factor μ by computing majority value for blocks of length μ, set bit to 1 in draws for even μ (no average effect as $x^\star = 0101\ldots$).
- Fixed-Length Search Space: bit string length now $\mu * n$.



1

original bit string (here: variable-length)

x 0101 0110 0000 1110 1000 0

2

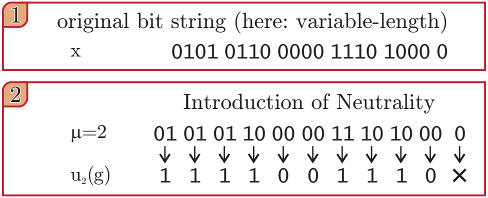
Introduction of Neutrality

μ=2 01 01 01 10 00 00 11 10 10 00 0

$u_2(g)$ 1 1 1 1 0 0 1 1 1 0 ✕

Neutrality Layer

- The application of a search operator is *neutral* if it yields no change in objective value [15, 16].
- Usually negative impact on performance [17, 18].
- Transformation $u(x)$ shortens x by factor μ by computing majority value for blocks of length μ, set bit to 1 in draws for even μ (no average effect as $x^\star = \texttt{0101}\ldots$).
- Fixed-Length Search Space: bit string length now $\mu * n$.
- Variable-Length Search Space: if $l(x)$ no multiple of μ, ignore last $l(x) \bmod \mu$ bits.



1 original bit string (here: variable-length)

x 0101 0110 0000 1110 1000 0

2 Introduction of Neutrality

μ=2 01 01 01 10 00 00 11 10 10 00 0

$u_2(g)$ 1 1 1 1 0 0 1 1 1 0 ✕

Epistasis Layer

- The interaction between biological genes is epistatic if the effect on the fitness from altering one gene depends on the allelic state of other genes [19, 20].

Epistasis Layer

- Two decision variables (here: bits) interact epistatically, if the contribution of one of these variables to the objective value depends on the value of the other variable [9, 20–22].

Epistasis Layer

- Two decision variables (here: bits) interact epistatically, if the contribution of one of these variables to the objective value depends on the value of the other variable [9, 20–22].
- *W-Model*: Bijective function e_ν translates a bit string x of length ν to a bit string $e_\nu(x)$ of the same length in $\mathcal{O}\left(\nu^2\right)$ steps.

Epistasis Layer

- Two decision variables (here: bits) interact epistatically, if the contribution of one of these variables to the objective value depends on the value of the other variable [9, 20–22].
- *W-Model*: Bijective function e_ν translates a bit string x of length ν to a bit string $e_\nu(x)$ of the same length in $\mathcal{O}\left(\nu^2\right)$ steps.

$$h(x_1, x_2) = 1 \Rightarrow h(e_\nu(x_1), e_\nu(x_2)) \geq \nu - 1 \; \forall x_1, x_2 \in \{0,1\}^\nu \qquad (1)$$

Epistasis Layer

- Two decision variables (here: bits) interact epistatically, if the contribution of one of these variables to the objective value depends on the value of the other variable [9, 20–22].
- *W-Model*: Bijective function e_ν translates a bit string x of length ν to a bit string $e_\nu(x)$ of the same length in $\mathcal{O}\left(\nu^2\right)$ steps.

$$h(x_1, x_2) = 1 \Rightarrow h(e_\nu(x_1), e_\nu(x_2)) \geq \nu - 1 \; \forall x_1, x_2 \in \{0, 1\}^\nu \qquad (1)$$

- A change of one bit in a bit string x leads to the change of at least $\nu - 1$ bits in the corresponding mapping $e_\nu(x)$.

Epistasis Layer

- Two decision variables (here: bits) interact epistatically, if the contribution of one of these variables to the objective value depends on the value of the other variable [9, 20–22].
- *W-Model*: Bijective function e_ν translates a bit string x of length ν to a bit string $e_\nu(x)$ of the same length in $\mathcal{O}\left(\nu^2\right)$ steps.

$$h(x_1, x_2) = 1 \Rightarrow h(e_\nu(x_1), e_\nu(x_2)) \geq \nu - 1 \ \forall x_1, x_2 \in \{0,1\}^\nu \tag{1}$$

- A change of one bit in a bit string x leads to the change of at least $\nu - 1$ bits in the corresponding mapping $e_\nu(x)$.

$$e_\nu(x) = \begin{cases} e_\nu(x)[i] = \bigotimes_{\substack{\forall\ j \in \mathbb{N}_0 : 0 \leq j < \nu, \\ j \neq (i-1) \bmod \nu}} x[j] & \forall\ x : 0 \leq x < 2^{\nu-1} \\ \overline{e_\nu(x - 2^{\nu-1})} & \textit{otherwise} \end{cases} \tag{2}$$

Epistasis Layer

- Two decision variables (here: bits) interact epistatically, if the contribution of one of these variables to the objective value depends on the value of the other variable [9, 20–22].
- *W-Model*: Bijective function e_ν translates a bit string x of length ν to a bit string $e_\nu(x)$ of the same length in $\mathcal{O}\left(\nu^2\right)$ steps.

$$h(x_1, x_2) = 1 \Rightarrow h(e_\nu(x_1), e_\nu(x_2)) \geq \nu - 1 \;\forall x_1, x_2 \in \{0,1\}^\nu \tag{1}$$

- A change of one bit in a bit string x leads to the change of at least $\nu - 1$ bits in the corresponding mapping $e_\nu(x)$.

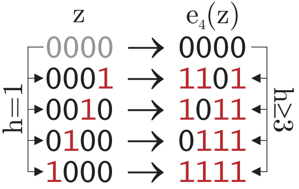
$$e_\nu(x) = \begin{cases} e_\nu(x)[i] = \bigotimes\limits_{\substack{\forall\, j \in \mathbb{N}_0 : 0 \leq j < \nu, \\ j \neq (i-1) \bmod \nu}} x[j] & \forall\, x : 0 \leq x < 2^{\nu-1} \\ \overline{e_\nu(x - 2^{\nu-1})} & \textit{otherwise} \end{cases} \tag{2}$$

- Each bit in x influences the value of $\nu - 1$ bits in $e_\nu(x)$.

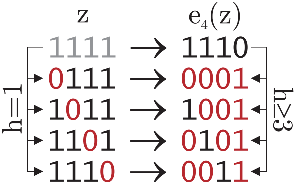
Epistasis Layer

$$e_\nu(x) = \begin{cases} e_\nu(x)[i] = \bigotimes\limits_{\substack{\forall\ j\in\mathbb{N}_0:0\le j<\nu,\\ j\neq(i-1) \bmod \nu}} x[j] & \forall\ x : 0 \le x < 2^{\nu-1} \\ \overline{e_\nu(x-2^{\nu-1})} & \textit{otherwise} \end{cases} \tag{2}$$

- Each bit in x influences the value of $\nu - 1$ bits in $e_\nu(x)$.



z		$e_4(z)$
0000	$\longrightarrow$	0000
0001	$\longrightarrow$	1101
0010	$\longrightarrow$	1011
0100	$\longrightarrow$	0111
1000	$\longrightarrow$	1111



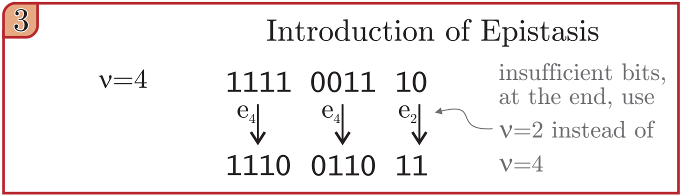
z		$e_4(z)$
1111	$\longrightarrow$	1110
0111	$\longrightarrow$	0001
1011	$\longrightarrow$	1001
1101	$\longrightarrow$	0101
1110	$\longrightarrow$	0011

z		$e_4(z)$
0011	$\longrightarrow$	0110
0101	$\longrightarrow$	1010
0110	$\longrightarrow$	1100
1001	$\longrightarrow$	0010
1010	$\longrightarrow$	0100
1100	$\longrightarrow$	1000

Epistasis Layer

$$e_\nu(x) = \begin{cases} e_\nu(x)[i] = \bigotimes\limits_{\substack{\forall\, j\in\mathbb{N}_0:0\le j<\nu,\\ j\ne(i-1) \bmod \nu}} x[j] & \forall\, x: 0\le x<2^{\nu-1}\\ \overline{e_\nu(x-2^{\nu-1})} & \textit{otherwise}\end{cases} \tag{2}$$

- Each bit in x influences the value of $\nu-1$ bits in $e_\nu(x)$.
- Candidate solutions divided into blocks of length ν to be transformed separately, if block of length $l(x) \bmod \nu$ remains, it is transformed with $e_{l(x) \bmod \nu}$.



3

Introduction of Epistasis

ν=4

1111 0011 10

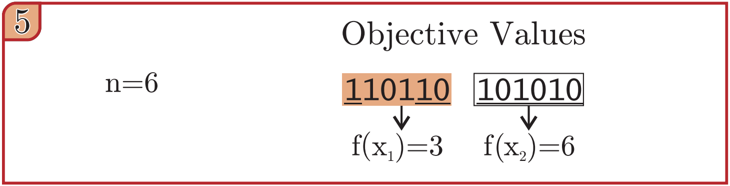
e4 e4 e2

1110 0110 11

insufficient bits, at the end, use ν=2 instead of ν=4

Multi-Objectivity Layer

- Many optimization problems are multi-objective, i.e., involve multiple, possible conflicting criteria [23–25]



5

Objective Values

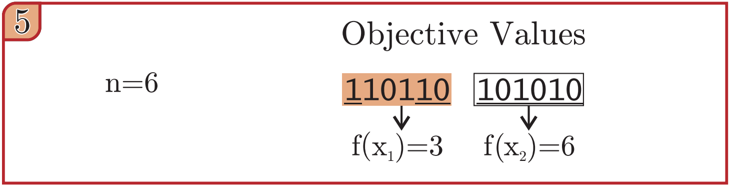
n=6

$\underline{1}101\underline{10}$ $\boxed{\underline{101010}}$

$f(x_1)=3$ $f(x_2)=6$

Multi-Objectivity Layer

- Many optimization problems are multi-objective, i.e., involve multiple, possible conflicting criteria [23–25]
- We simply interleave m instances of the *W-Model* to get an m-objective problem.


5

Objective Values

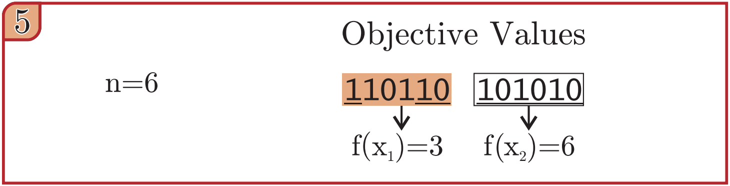
n=6

110110 101010

$f(x_1)=3$ $f(x_2)=6$

Multi-Objectivity Layer

- Many optimization problems are multi-objective, i.e., involve multiple, possible conflicting criteria [23–25]
- We simply interleave m instances of the *W-Model* to get an m-objective problem.
- Disagreement in the orthogonal objective functions can be simulated via epistasis $\nu > 2$.



Ruggedness and Deceptiveness Layer

- Strong causality means that small changes in a candidate solution lead to small changes in the objective value [26, 27].

Ruggedness and Deceptiveness Layer

- Strong causality means that small changes in a candidate solution lead to small changes in the objective value [26, 27].
- Rugged fitness landscape: small changes in candidate solution $\Longrightarrow$ large changes in objective value.

Ruggedness and Deceptiveness Layer

- Strong causality means that small changes in a candidate solution lead to small changes in the objective value [26, 27].
- Rugged fitness landscape: small changes in candidate solution $\Longrightarrow$ large changes in objective value.
- Deceptive fitness landscape: following changes towards declining objective function leads aways from optimum.

Ruggedness and Deceptiveness Layer

- Strong causality means that small changes in a candidate solution lead to small changes in the objective value [26, 27].
- Rugged fitness landscape: small changes in candidate solution $\Longrightarrow$ large changes in objective value.
- Deceptive fitness landscape: following changes towards declining objective function leads aways from optimum.
- Epistasis is a source of ruggedness and deceptiveness.

Ruggedness and Deceptiveness Layer

- Strong causality means that small changes in a candidate solution lead to small changes in the objective value [26, 27].
- Rugged fitness landscape: small changes in candidate solution $\Longrightarrow$ large changes in objective value.
- Deceptive fitness landscape: following changes towards declining objective function leads aways from optimum.
- Epistasis is a source of ruggedness and deceptiveness.
- For $\nu = 1$ in *W-Model*: Change one bit in x leads to change of 1 in objective value.

Ruggedness and Deceptiveness Layer

- Strong causality means that small changes in a candidate solution lead to small changes in the objective value [26, 27].
- Rugged fitness landscape: small changes in candidate solution $\Longrightarrow$ large changes in objective value.
- Deceptive fitness landscape: following changes towards declining objective function leads aways from optimum.
- Epistasis is a source of ruggedness and deceptiveness.
- For $\nu = 1$ in *W-Model*: Change one bit in x leads to change of 1 in objective value.
- If we improve candidate solution from worst possible to best bit-by-bit, we traverse objective values $(n, n-1, \ldots, 2, 1, 0)$.

Ruggedness and Deceptiveness Layer

- Strong causality means that small changes in a candidate solution lead to small changes in the objective value [26, 27].
- Rugged fitness landscape: small changes in candidate solution $\Longrightarrow$ large changes in objective value.
- Deceptive fitness landscape: following changes towards declining objective function leads aways from optimum.
- Epistasis is a source of ruggedness and deceptiveness.
- For $\nu = 1$ in *W-Model*: Change one bit in x leads to change of 1 in objective value.
- If we improve candidate solution from worst possible to best bit-by-bit, we traverse objective values $(n, n-1, \ldots, 2, 1, 0)$.
- If we can exchange the values in this sequence, this will increase the ruggedness or cause deceptiveness!

Ruggedness and Deceptiveness Layer

- Strong causality means that small changes in a candidate solution lead to small changes in the objective value [26, 27].
- Rugged fitness landscape: small changes in candidate solution $\Longrightarrow$ large changes in objective value.
- Deceptive fitness landscape: following changes towards declining objective function leads aways from optimum.
- Epistasis is a source of ruggedness and deceptiveness.
- For $\nu = 1$ in *W-Model*: Change one bit in x leads to change of 1 in objective value.
- If we improve candidate solution from worst possible to best bit-by-bit, we traverse objective values $(n, n-1, \dots, 2, 1, 0)$.
- If we can exchange the values in this sequence, this will increase the ruggedness or cause deceptiveness!
- The ruggedness Δ of a permutation r be $\Delta(r) = \sum_{i=0}^{n-1} |r_i - r_{i+1}|$.

Ruggedness and Deceptiveness Layer

- If we improve candidate solution from worst possible to best bit-by-bit, we traverse objective values $(n, n-1, \ldots, 2, 1, 0)$.
- The ruggedness Δ of a permutation r be $\Delta(r) = \sum_{i=0}^{n-1} |r_i - r_{i+1}|$.

Ruggedness and Deceptiveness Layer

- If we improve candidate solution from worst possible to best bit-by-bit, we traverse objective values $(n, n-1, \ldots, 2, 1, 0)$.
- The ruggedness Δ of a permutation r be $\Delta(r) = \sum_{i=0}^{n-1} |r_i - r_{i+1}|$.
- Original sequence above has $\Delta(n \ldots 0) = n$ and maximum possible value is $\hat{\Delta} = \frac{n(n+1)}{2}$.

Ruggedness and Deceptiveness Layer

- If we improve candidate solution from worst possible to best bit-by-bit, we traverse objective values $(n, n-1, \dots, 2, 1, 0)$.
- The ruggedness Δ of a permutation r be $\Delta(r) = \sum_{i=0}^{n-1} |r_i - r_{i+1}|$.
- Original sequence above has $\Delta(n \dots 0) = n$ and maximum possible value is $\hat{\Delta} = \frac{n(n+1)}{2}$.
- We define mappings based on permutations $r_{\gamma'}$ with the following features

Ruggedness and Deceptiveness Layer

- If we improve candidate solution from worst possible to best bit-by-bit, we traverse objective values $(n, n-1, \dots, 2, 1, 0)$.
- The ruggedness Δ of a permutation r be $\Delta(r) = \sum_{i=0}^{n-1} |r_i - r_{i+1}|$.
- Original sequence above has $\Delta(n \dots 0) = n$ and maximum possible value is $\hat{\Delta} = \frac{n(n+1)}{2}$.
- We define mappings based on permutations $r_{\gamma'}$ with the following features:
  1. They are bijective (since they are permutations).

Ruggedness and Deceptiveness Layer

- If we improve candidate solution from worst possible to best bit-by-bit, we traverse objective values $(n, n-1, \dots, 2, 1, 0)$.
- The ruggedness Δ of a permutation r be $\Delta(r) = \sum_{i=0}^{n-1} |r_i - r_{i+1}|$.
- Original sequence above has $\Delta(n \dots 0) = n$ and maximum possible value is $\hat{\Delta} = \frac{n(n+1)}{2}$.
- We define mappings based on permutations $r_{\gamma'}$ with the following features:
  1. They are bijective (since they are permutations).
  2. They must preserve the optimal value, i.e., $r_{\gamma'}[0] = 0$.

Ruggedness and Deceptiveness Layer

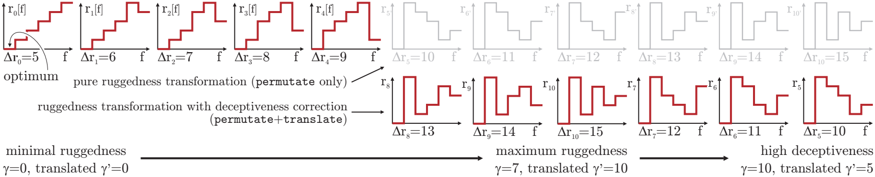
- If we improve candidate solution from worst possible to best bit-by-bit, we traverse objective values $(n, n-1, \dots, 2, 1, 0)$.
- The ruggedness Δ of a permutation r be $\Delta(r) = \sum_{i=0}^{n-1} |r_i - r_{i+1}|$.
- Original sequence above has $\Delta(n \dots 0) = n$ and maximum possible value is $\hat{\Delta} = \frac{n(n+1)}{2}$.
- We define mappings based on permutations $r_{\gamma'}$ with the following features:
  1. They are bijective (since they are permutations).
  2. They must preserve the optimal value, i.e., $r_{\gamma'}[0] = 0$.
  3. $\Delta(r_{\gamma'}) = n + \gamma'$.

Ruggedness and Deceptiveness Layer

- If we improve candidate solution from worst possible to best bit-by-bit, we traverse objective values $(n, n-1, \ldots, 2, 1, 0)$.
- The ruggedness Δ of a permutation r be $\Delta(r) = \sum_{i=0}^{n-1} |r_i - r_{i+1}|$.
- Original sequence above has $\Delta(n \ldots 0) = n$ and maximum possible value is $\hat{\Delta} = \frac{n(n+1)}{2}$.
- We define mappings based on permutations $r_{\gamma'}$ with the following features:
  1. They are bijective (since they are permutations).
  2. They must preserve the optimal value, i.e., $r_{\gamma'}[0] = 0$.
  3. $\Delta(r_{\gamma'}) = n + \gamma'$.
- With $\gamma' \in 0 \ldots (\hat{\Delta} - n)$, we can fine-tune the ruggedness.

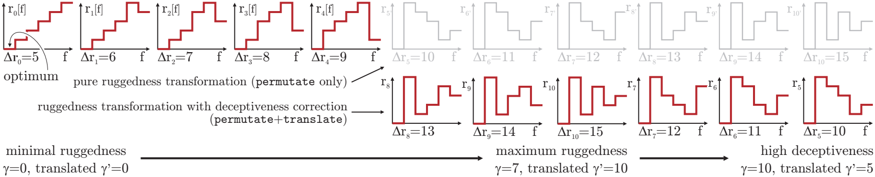
Ruggedness and Deceptiveness Layer

- Such permutations can be generated using the algorithm defined in [1]



Ruggedness and Deceptiveness Layer

- Such permutations can be generated using the algorithm defined in [1]
- Original algorithm produces alternating sequences of rugged and deceptive problems, the latter are much harder.



pure ruggedness transformation (`permutate` only)

ruggedness transformation with deceptiveness correction (`permutate`+`translate`)

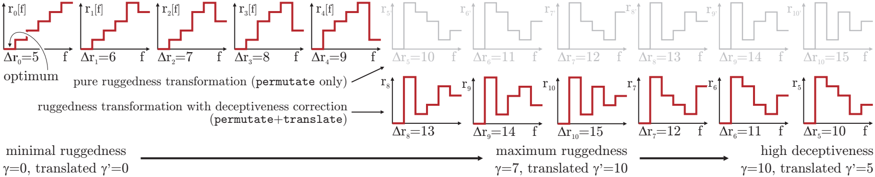
minimal ruggedness γ=0, translated γ'=0

maximum ruggedness γ=7, translated γ'=10

high deceptiveness γ=10, translated γ'=5

Ruggedness and Deceptiveness Layer

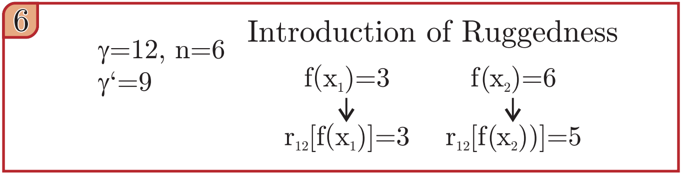
- Such permutations can be generated using the algorithm defined in [1]
- Original algorithm produces alternating sequences of rugged and deceptive problems, the latter are much harder.
- Re-arrangement into permutations $r_{\gamma'}$ which nicely blend from smooth to rugged to deceptive problems.



r0[f]
r1[f]
r2[f]
r3[f]
r4[f]
r5'
r6'
r7'
r8'
r9'
r10'
Δr0=5 f
Δr1=6 f
Δr2=7 f
Δr3=8 f
Δr4=9 f
Δr5=10 f
Δr6=11 f
Δr7=12 f
Δr8=13 f
Δr9=14 f
Δr10=15 f
optimum
pure ruggedness transformation (permutate only)
ruggedness transformation with deceptiveness correction (permutate+translate)
r8
r9
r10
r7
r6
r5
Δr8=13
Δr9=14 f
Δr10=15
Δr7=12 f
Δr6=11 f
Δr5=10 f
minimal ruggedness γ=0, translated γ'=0
maximum ruggedness γ=7, translated γ'=10
high deceptiveness γ=10, translated γ'=5

Ruggedness and Deceptiveness Layer

- Such permutations can be generated using the algorithm defined in [1]
- Original algorithm produces alternating sequences of rugged and deceptive problems, the latter are much harder.
- Re-arrangement into permutations $r_{\gamma'}$ which nicely blend from smooth to rugged to deceptive problems.
- Permutation serves as lookup-table to map objective values.



6

γ=12, n=6

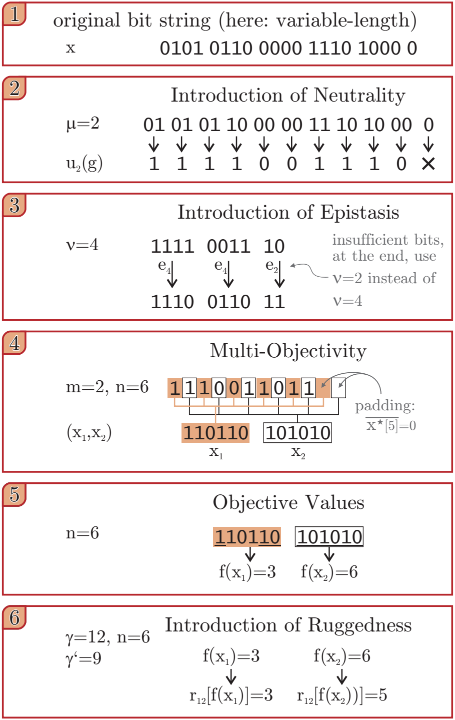
γ‘=9

Introduction of Ruggedness

$$\mathrm{f}(\mathrm{x}_1)=3 \downarrow \mathrm{r}_{12}[\mathrm{f}(\mathrm{x}_1)]=3$$

$$\mathrm{f}(\mathrm{x}_2)=6 \downarrow \mathrm{r}_{12}[\mathrm{f}(\mathrm{x}_2))]=5$$

Summary



1 original bit string (here: variable-length)

x 0101 0110 0000 1110 1000 0

2 Introduction of Neutrality

$\mu=2$ 01 01 01 10 00 00 11 10 10 00 0

$u_2(g)$ 1 1 1 1 0 0 1 1 1 0 ×

3 Introduction of Epistasis

$\nu=4$ 1111 0011 10

e_4 e_4 e_2

1110 0110 11

insufficient bits, at the end, use $\nu=2$ instead of $\nu=4$

4 Multi-Objectivity

$m=2$, $n=6$ 1 1 1 0 0 1 1 0 1 1

padding: $\overline{x^{\star}}[5]=0$

(x_1,x_2) 110110 101010

x_1 x_2

5 Objective Values

$n=6$ 110110 101010

$f(x_1)=3$ $f(x_2)=6$

6 Introduction of Ruggedness

$\gamma=12$, $n=6$

$\gamma'=9$

$f(x_1)=3$ $f(x_2)=6$

$r_{12}[f(x_1)]=3$ $r_{12}[f(x_2))]=5$

Section Outline

These are the slides for paper [1] presented at the BB-DOB workshop at GECCO'2018.

1. Introduction
2. The *W-Model*
3. Experiment
4. Summary

website

Experiments

- Extensive experiments for the variable-length representation with multi-objectivity were performed in [28] and we use this data here.

Experiments

- Extensive experiments for the variable-length representation with multi-objectivity were performed in [28] and we use this data here.
- Now, a much bigger dataset for the fixed-length representation and a single-objective $(m = 1)$ is available in [29].

Experiments

- Extensive experiments for the variable-length representation with multi-objectivity were performed in [28] and we use this data here.
- Now, a much bigger dataset for the fixed-length representation and a single-objective $(m = 1)$ is available in [29].
- In the experiment discussed here [28], we apply

Experiments

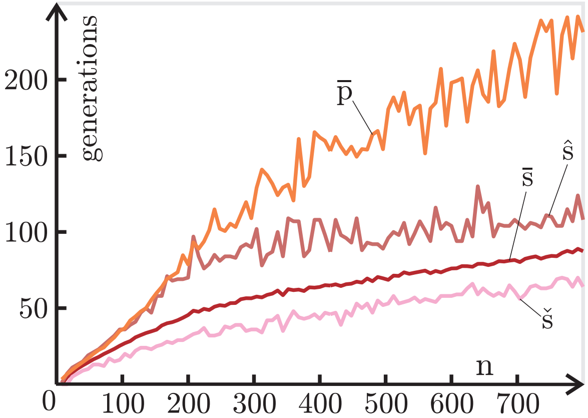
- Extensive experiments for the variable-length representation with multi-objectivity were performed in [28] and we use this data here.
- Now, a much bigger dataset for the fixed-length representation and a single-objective $(m = 1)$ is available in [29].
- In the experiment discussed here [28], we apply
  - a standard multi-objective genetic algorithm with
  - population size 1000,
  - single-point crossover,
  - single-bit mutation,
  - tournament selection with tournament size 5,
  - Pareto ranking, and a
  - variable-length bit string genome with a maximum string length of 8000 bits.

Experiments

- Extensive experiments for the variable-length representation with multi-objectivity were performed in [28] and we use this data here.
- Now, a much bigger dataset for the fixed-length representation and a single-objective $(m = 1)$ is available in [29].
- In the experiment discussed here [28], we apply
    - a standard multi-objective genetic algorithm with
    - population size 1000,
    - single-point crossover,
    - single-bit mutation,
    - tournament selection with tournament size 5,
    - Pareto ranking, and a
    - variable-length bit string genome with a maximum string length of 8000 bits.
- We distinguish *success* (after s generations), i.e., finding a string x with $f(x) = 0$ (but which may be too long) and *perfection*, i.e., finding $x^{\star}$ (after $p \geq s$ generations).

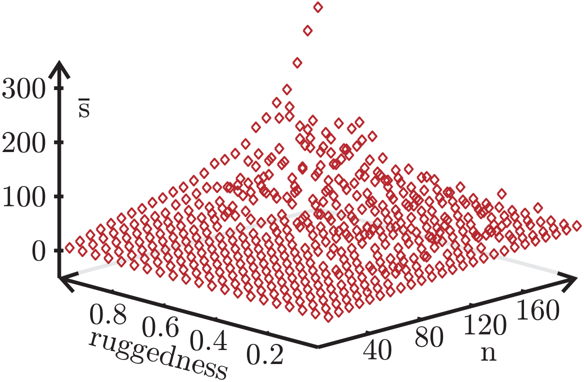
Basic Problem n

- The minimum, average, and maximum success generations $\check{s}$, $\overline{s}$, and $\hat{s}$ measured rise almost linearly after the basic problem parameter n has exceeded 300 bits.



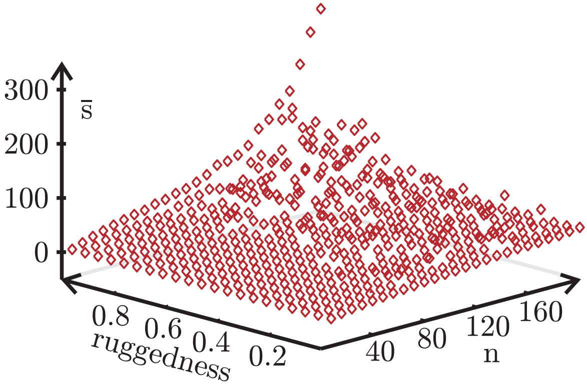
Ruggedness + Basic Problem n

- Ruggedness parameter normalized into the range $[0, 1]$, because its range depends on n.



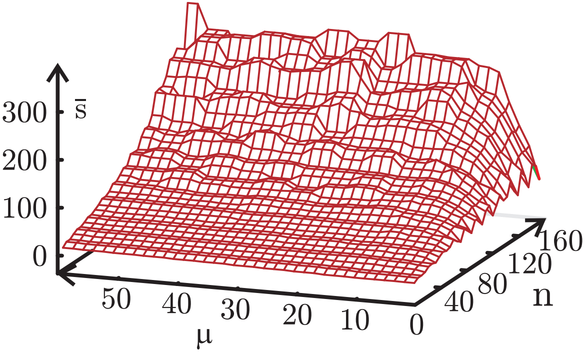
Ruggedness + Basic Problem n

- Ruggedness parameter normalized into the range $[0, 1]$, because its range depends on n.
- Apart from a few peaks in the diagram occurring for $n > 70$, the problem hardness, as expected, increases very fast with the ruggedness.



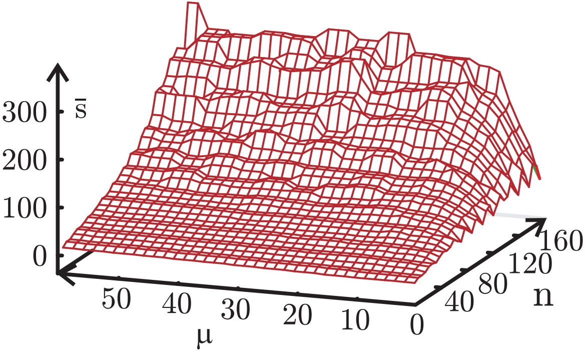
Neutrality μ + Basic Problem n

- Until a degree of $\mu \approx 10$, the problems rapidly gets harder with rising redundancy μ.



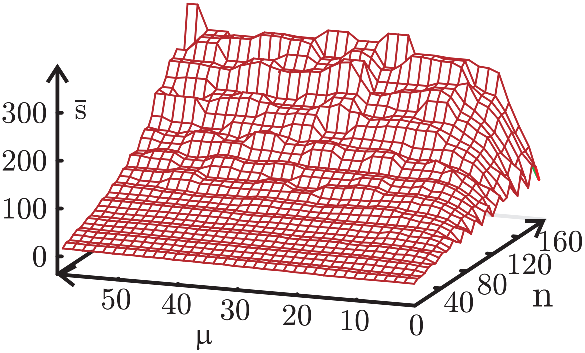
Neutrality μ + Basic Problem n

- Until a degree of $\mu \approx 10$, the problems rapidly gets harder with rising redundancy μ.
- From there on, a further increase of μ only leads to a very slow increase in hardness.



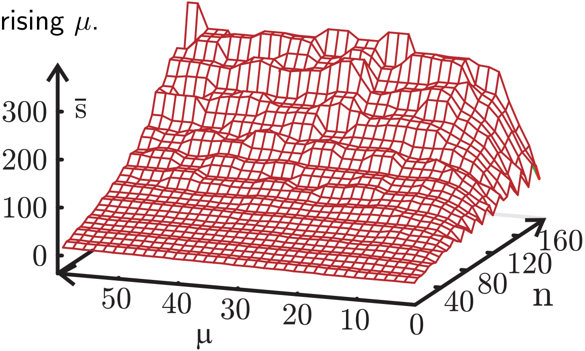
Neutrality μ + Basic Problem n

- Until a degree of $\mu \approx 10$, the problems rapidly gets harder with rising redundancy μ.
- From there on, a further increase of μ only leads to a very slow increase in hardness.
- Probably cause: for crossover, larger μ make no big difference.



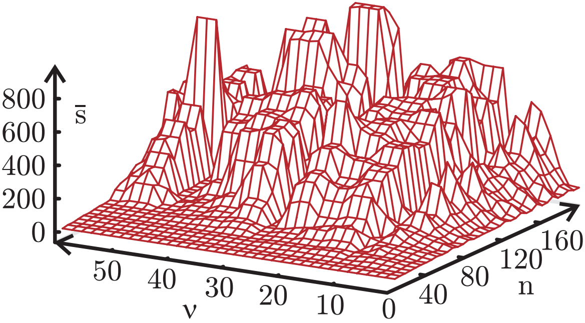
Neutrality μ + Basic Problem n

- Until a degree of $\mu \approx 10$, the problems rapidly gets harder with rising redundancy μ.
- From there on, a further increase of μ only leads to a very slow increase in hardness.
- Probably cause: for crossover, larger μ make no big difference.
- Experiments with lower crossover rate led to quick decrease of performance for rising μ.



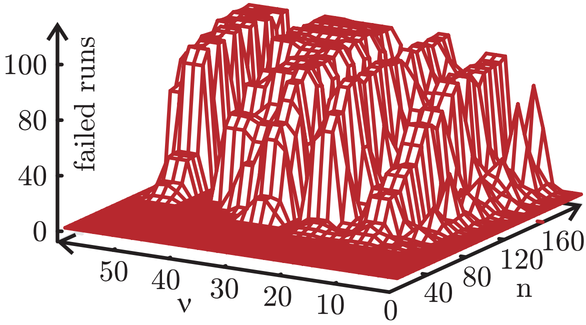
Epistasis ν + Basic Problem n

- Problem complexity steeply increases with rising epistasis (values of ν).



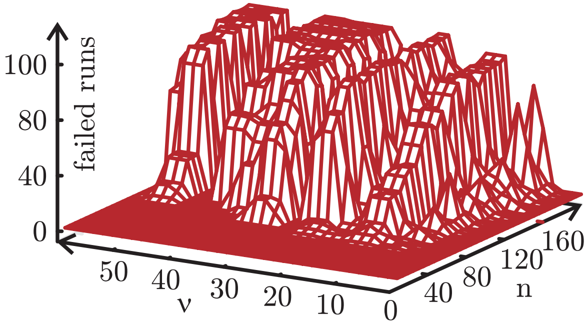
Epistasis ν + Basic Problem n

- Problem complexity steeply increases with rising epistasis (values of ν).
- Number of runs that cannot solve problem in 1000 generations quickly rises with ν.



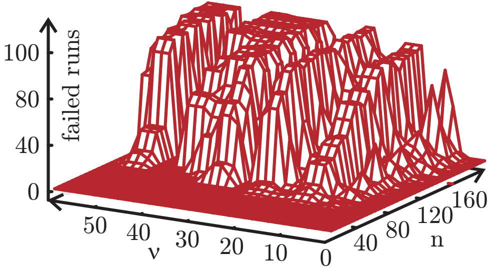
Epistasis ν + Basic Problem n

- Problem complexity steeply increases with rising epistasis (values of ν).
- Number of runs that cannot solve problem in 1000 generations quickly rises with ν.
- Problems for which $\nu = 2 + 4v : v \in \mathbb{N}$ are unexpectedly easy.



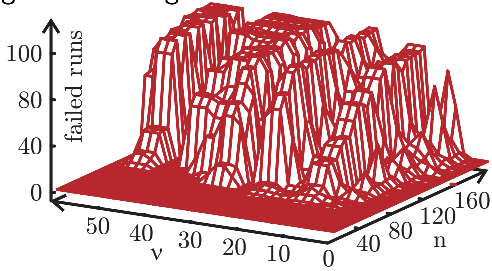
Epistasis ν + Basic Problem n

- Problem complexity steeply increases with rising epistasis (values of ν).
- Number of runs that cannot solve problem in 1000 generations quickly rises with ν.
- Problems for which $\nu = 2 + 4v : v \in \mathbb{N}$ are unexpectedly easy.
- The epistasis mapping e_ν decreases the Hamming distance for x_1, x_2 which have originally $h(x_1, x_2) = \nu/2$ in such cases [28].



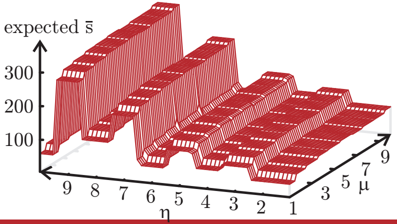
Epistasis ν + Basic Problem n

- Problem complexity steeply increases with rising epistasis (values of ν).
- Number of runs that cannot solve problem in 1000 generations quickly rises with ν.
- Problems for which $\nu = 2 + 4v : v \in \mathbb{N}$ are unexpectedly easy.
- The epistasis mapping e_ν decreases the Hamming distance for x_1, x_2 which have originally $h(x_1, x_2) = \nu/2$ in such cases [28].
- We suggest not using these settings.



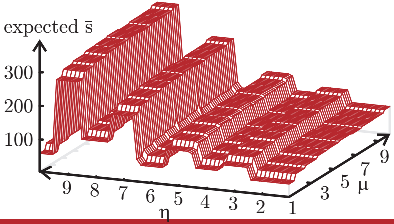
Expected Epistasis ν + Neutrality μ

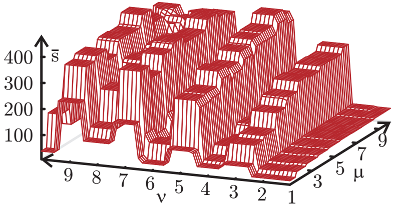
- Two separate experiments with either neutrality or epistasis added together: “expected $\overline{s}$”



Expected Epistasis ν + Neutrality μ

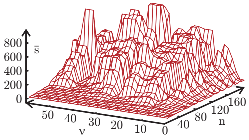
- Two separate experiments with either neutrality or epistasis added together: "expected $\overline{s}$"
- Actual experiment with both neutrality and epistasis: shape similar to expectation, but 100 generations harder

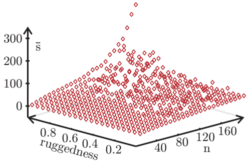


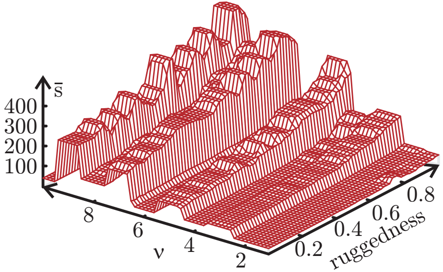


Real Epistasis ν + Ruggedness

- If both epistasis and ruggedness are used, results look similar to expectations of adding the results with only ruggedness to those with only epistasis.

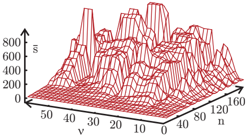


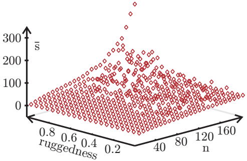


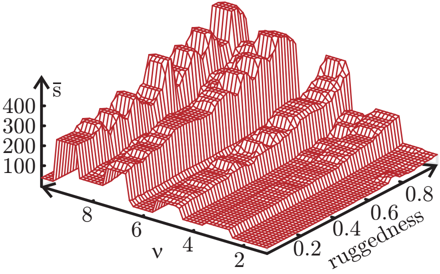


Real Epistasis ν + Ruggedness

- If both epistasis and ruggedness are used, results look similar to expectations of adding the results with only ruggedness to those with only epistasis.
- A rising ruggedness component leads, however, to over-proportional increases in $\overline{s}$.







Section Outline

These are the slides for paper [1] presented at the BB-DOB workshop at GECCO'2018.

1 Introduction

2 The *W-Model*

3 Experiment

4 Summary

website

Summary

- We have discussed several requirements for good discrete optimization benchmark problems.

Summary

- We have discussed several requirements for good discrete optimization benchmark problems.
- We have introduced the *W-Model* as an example problem that fulfills these requirements and suggest it for inclusion in the BB-DOB problem suite.

Summary

- We have discussed several requirements for good discrete optimization benchmark problems.
- We have introduced the *W-Model* as an example problem that fulfills these requirements and suggest it for inclusion in the BB-DOB problem suite.
- It can be applied with a fixed-length and a variable-length represention.

Summary

- We have discussed several requirements for good discrete optimization benchmark problems.
- We have introduced the *W-Model* as an example problem that fulfills these requirements and suggest it for inclusion in the BB-DOB problem suite.
- It can be applied with a fixed-length and a variable-length represention.
- Our old experiments with the variable-length representation show that it behaves well and as expected.

Summary

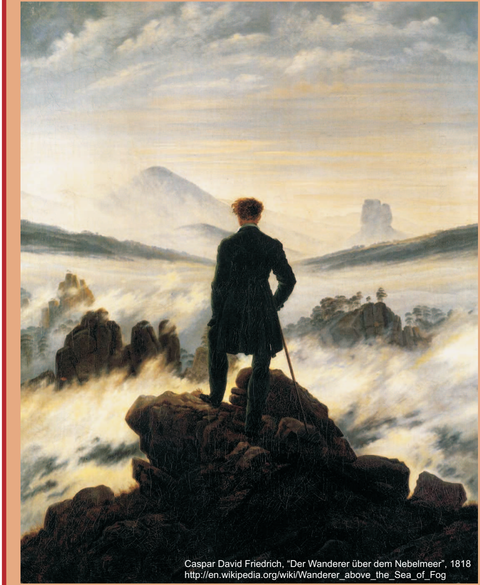
- We have discussed several requirements for good discrete optimization benchmark problems.
- We have introduced the *W-Model* as an example problem that fulfills these requirements and suggest it for inclusion in the BB-DOB problem suite.
- It can be applied with a fixed-length and a variable-length represention.
- Our old experiments with the variable-length representation show that it behaves well and as expected.
- A new implementation [10] is provided along with new results for the fixed-length representation in [29].

谢谢

Thank you

Thomas Weise & Zijun Wu [汤卫思 & 吴自军]
tweise@hfuu.edu.cn
http://iao.hfuu.edu.cn

Hefei University, South Campus 2
Institute of Applied Optimization
Shushan District, Hefei, Anhui, China



Caspar David Friedrich, “Der Wanderer über dem Nebelmeer”, 1818
http://en.wikipedia.org/wiki/Wanderer_above_the_Sea_of_Fog

Bibliography

Bibliography I

1. Thomas Weise and Zijun Wu. Difficult features of combinatorial optimization problems and the tunable *W-Model* benchmark problem for simulating them. In *Black Box Discrete Optimization Benchmarking (BB-DOB) Workshop in Companion Material Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2018), July 15–19, 2018, Kyoto, Japan*, pages 1769–1776. ACM. doi: 10.1145/3205651.3208240. ISBN: 978-1-4503-5764-7. Additional experimental results with the *W-Model*, which were not used in this paper, can be found in [29] at doi:10.5281/zenodo.1256883 based on implementation [10].
2. Thomas Weise, Raymond Chiong, and Ke Tang. Evolutionary optimization: Pitfalls and booby traps. *Journal of Computer Science and Technology (JCST)*, 27:907–936, 2012. doi: 10.1007/s11390-012-1274-4.
3. Thomas Weise, Michael Zapf, Raymond Chiong, and Antonio Jesús Nebro Urbaneja. Why is optimization difficult? In Raymond Chiong, editor, *Nature-Inspired Algorithms for Optimisation*, volume 193 of *Studies in Computational Intelligence*, chapter 1, pages 1–50. Springer-Verlag, Berlin/Heidelberg, 2009. ISBN 978-3-642-00266-3. doi: 10.1007/978-3-642-00267-0_1.
4. David Lee Applegate, Robert E. Bixby, Vašek Chvátal, and William John Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, Princeton, NJ, USA, 2007.
5. Thomas Weise, Raymond Chiong, Ke Tang, Jörg Lässig, Shigeyoshi Tsutsui, Wenxiang Chen, Zbigniew Michalewicz, and Xin Yao. Benchmarking optimization algorithms: An open source framework for the traveling salesman problem. *IEEE Computational Intelligence Magazine (CIM)*, 9(3):40–52, August 2014. doi: 10.1109/MCI.2014.2326101.
6. *Proc. of the 18th Intl. Conf. on Theory and Applications of Satisfiability Testing (SAT 2015)*, volume 9340 of *Lecture Notes in Computer Science book series (LNCS)*, Cham, September 24-27, 2015. Springer. ISBN 978-3-319-24317-7. doi: 10.1007/978-3-319-24318-4.
7. Holger H. Hoos and Thomas Stützle. SATLIB: An online resource for research on SAT. In *SAT2000 – Highlights of Satisfiability Research in the Year 2000*, volume 63 of *Frontiers in Artificial Intelligence and Applications*, pages 283–292, Amsterdam, The Netherlands, 2000. IOS Press. ISBN 978-1-58603-061-2. URL `http://www.cs.ubc.ca/~hoos/Publ/sat2000-satlib.pdf`.
8. Thomas Weise, Stefan Niemczyk, Hendrik Skubch, Roland Reichle, and Kurt Geihs. A tunable model for multi-objective, epistatic, rugged, and neutral fitness landscapes. In *Genetic and Evolutionary Computation Conference*, pages 795–802, Atlanta, GA, USA, 2008. ACM Press.
9. Thomas Weise. *Global Optimization Algorithms – Theory and Application*. it-weise.de (self-published), Germany, 2009. URL `http://www.it-weise.de/projects/book.pdf`.

Bibliography II

10. Thomas Weise. The *W-Model*, a tunable black-box discrete optimization benchmarking (bb-dob) problem, implemented for the bb-dob@gecco workshop, June 2018. URL `http://github.com/thomasWeise/BBDOB_W_Model`. Open source implementation *W-Model* [1] provided in a GitHub repository, used in [29].
11. David H. Ackley. *A Connectionist Machine for Genetic Hillclimbing*. PhD thesis, Carnegy Mellon University (CMU), Pittsburgh, PA, USA, 1987.
12. Dirk Thierens and David Edward Goldberg. Convergence models of genetic algorithm selection schemes. In *Proc. of the Third Conf. on Parallel Problem Solving from Nature (PPSN III)*, volume 866/1994 of *Lecture Notes in Computer Science (LNCS)*, pages 119–129, Berlin, Germany, October 9–14, 1994. Springer-Verlag GmbH. ISBN 0387584846. doi: 10.1007/3-540-58484-6_256.
13. Brad L. Miller and David Edward Goldberg. Genetic algorithms, selection schemes, and the varying effects of noise. *Evolutionary Computation*, 4(2):113–131, 1996. doi: 10.1162/evco.1996.4.2.113.
14. Tobias Blickle and Lothar Thiele. A comparison of selection schemes used in genetic algorithms. TIK-Report 11, ETH Zürich, Department of Electrical Engineering, Computer Engineering and Networks Laboratory (TIK), Zürich, Switzerland, December 1995. URL `ftp://ftp.tik.ee.ethz.ch/pub/publications/TIK-Report11.ps`.
15. Christian M. Reidys and Peter F. Stadler. Neutrality in fitness landscapes. *Journal of Applied Mathematics and Computation*, 117(2–3):321–350, 2001. doi: 10.1016/S0096-3003(99)00166-6.
16. Lionel Barnett. Ruggedness and neutrality – the nkp family of fitness landscapes. In *Proc. of the Sixth Intl. Conf. on Artificial Life (Artificial Life VI)*, volume 6 of *Complex Adaptive Systems*, pages 18–27, Cambridge, MA, USA, June 27–29, 1998. MIT Press. ISBN 0-262-51099-5.
17. Joshua D. Knowles and Richard A. Watson. On the utility of redundant encodings in mutation-based evolutionary search. In *Proc. of the 7th Intl. Conf. on Parallel Problem Solving from Nature (PPSN VII)*, volume 2439 of *Lecture Notes in Computer Science (LNCS)*, pages 88–98, Berlin, Germany, September 7–11, 2002. Springer-Verlag GmbH. ISBN 3-540-44139-5. doi: 10.1007/3-540-45712-7_9.
18. Franz Rothlauf. *Representations for Genetic and Evolutionary Algorithms*, volume 104 of *Studies in Fuzziness and Soft Computing*. Springer-Verlag, Berlin/Heidelberg, second edition, 2006. doi: 10.1007/3-540-32444-5.
19. Jay L. Lush. Progeny test and individual performance as indicators of an animal's breeding value. *Journal of Dairy Science (JDS)*, 18(1):1–19, 1935.
20. Lee Altenberg. Nk fitness landscapes. In Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors, *Handbook of Evolutionary Computation*, chapter B2.7.2. Oxford University Press, Oxford, England, UK, 1997.

Bibliography III

21. Yuval Davidor. Epistasis variance: A viewpoint on GA-hardness. In *Proc. of the First Workshop on Foundations of Genetic Algorithms (FOGA'90)*, pages 23–35, San Francisco, CA, USA, July 15–18, 1990. Morgan Kaufmann Publishers Inc. ISBN 1-55860-170-8.
22. Bart Naudts and Alain Verschoren. Epistasis on finite and infinite spaces. In *Proc. of the 8th Intl. Conf. on Systems Research, Informatics and Cybernetics (InterSymp'96)*, pages 19–23, Tecumseh, ON, Canada, August 14–18, 1996. Intl. Institute for Advanced Studies in Systems Research and Cybernetic (IIAS).
23. Kalyanmoy Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley Interscience Series in Systems and Optimization. John Wiley & Sons Ltd., New York, NY, USA, 2001. ISBN 047187339X.
24. Carlos Artemio Ceollo Coello. An updated survey of evolutionary multiobjective optimization techniques: State of the art and future trends. In *Congress on Evolutionary Computation (CEC)*, pages 3–13, Piscataway, NJ, USA, July 6–9, 1999. IEEE Press. doi: 10.1109/CEC.1999.781901.
25. Carlos M. Fonseca and Peter J. Fleming. Multiobjective optimization and multiple constraint handling with evolutionary algorithms – part i: A unified formulation. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, 28(1):26–37, 1998.
26. Charles Campbell Palmer and Aaron Kershenbaum. Representing trees in genetic algorithms. In *Proc. of the 1st IEEE Conf. on Evolutionary Comput. (CEC'94)*, volume 1, pages 379–384, Piscataway, NJ, USA, June 27–29, 1994. IEEE Comp. Soc. ISBN 0-7803-1899-4.
27. Charles Campbell Palmer. *An approach to a problem in network design using genetic algorithms*. PhD thesis, Polytechnic University, New York, NY, 1994.
28. Stefan Niemczyk. Ein benchmark problem für globale optimierungsverfahren. Bachelor's thesis, Distributed Systems Group, University of Kassel, May 2008. Supervisor: Thomas Weise.
29. Thomas Weise. Results for several simple algorithms on the *W-Model* for black-box discrete optimization benchmarking, June 2018. URL `https://doi.org/10.5281/zenodo.1256883`. Based on the implementation of *W-Model* [1] given in [10].