Introduction to Walsh Analysis

Alternative Views of the Genetic Algorithm

R. Paul Wiegand

paul@tesseract.org

ECLab

George Mason University

Outline of Discussion

Part I:	Overview of the Walsh Transform $\longleftarrow$
Part II:	Walsh Analysis of Fitness
Part III:	Walsh Analysis of Mixing Matrices
Part IV:	Conclusions

What is Walsh Analysis?

- Analysis (of a GA) using ***Walsh Transform***
- Historically mainly for landscape analysis
  - Way of "measuring where the energy/information content in a landscape is"
  - Helps define notions such as "building block" and "deception" more formally
- Recently applied to analysis of variation
  - Used in Vose dynamical systems model of SGA
  - Exposes properties of the *mixing matrix* in the SGA model

What is a Basis?

A set of linearly independent vectors in a vector space such that each vector in the space is a linear combination of vectors from the set.

What is a Basis?

A set of linearly independent vectors in a vector space such that each vector in the space is a linear combination of vectors from the set.

For Example:

Suppose U is the unit basis for $\Re^2$

$$U_1 = < 1 \ \ 0 > \qquad \vec{y} \in \Re^2$$
$$U_2 = < 0 \ \ 1 > \qquad \vec{y} = 2.0U_1 - 3.1U_2$$

What is a Basis?

A set of linearly independent vectors in a vector space such that each vector in the space is a linear combination of vectors from the set.

For Example:

Suppose U is the unit basis for $\Re^2$

$$\begin{aligned} U_1 &= <1 \;\; 0> \\ U_2 &= <0 \;\; 1> \end{aligned} \qquad\qquad \begin{aligned} \vec{y} &\in \Re^2 \\ \vec{y} &= 2.0U_1 - 3.1U_2 \end{aligned}$$

A basis can be seen as a type of *viewpoint* or *perspective*

Basis Transformations

- Takes a space and expresses it under a new/different basis
- $\vec{x} \mapsto W\vec{x}$ (assuming all objects are real)
- Change in viewpoint or perspective

What is the Walsh Transform?

- Discrete analog of the Fourier transform
- Transformation into the ***Walsh basis***
- Change in viewpoint:
  - For landscape analysis: to help see schema more clearly
  - For variation analysis: to help expose certain mathematical properties of the mixing matrix

What is the Walsh Transform?

- Discrete analog of the Fourier transform
- Transformation into the ***Walsh basis***
- Change in viewpoint:
  - For landscape analysis: to help see schema more clearly
  - For variation analysis: to help expose certain mathematical properties of the mixing matrix

For example:

Before: see points in landscape residing in implied partitions

Now: see schemata in landscape explicitly and points are implied by construction.

Outline of Discussion

Part I:	Overview of the Walsh Transform	$\surd$
Part II:	Walsh Analysis of Fitness	$\longleftarrow$
Part III:	Walsh Analysis of Mixing Matrices	
Part IV:	Conclusions	

Fitness & the Standard Basis

(Assuming binary representation)

- Indiv. are fixed-length bin. str.

  $$x \in \{0,1\}^\ell$$

- We can enumerate points and associated fitness values
- There is an *implied* basis:

$$f(j) = \sum_{i=00\ldots0}^{11\ldots1} f_i \delta_{ij}$$

  - Where $\delta_{ij} = 1$ when $i = j$ and 0 otherwise

Example:

3-bit landscape

Point	**Fitness**
000	f_{000}
001	f_{001}
$\vdots$	$\vdots$
111	f_{111}

Overview of Schema

- Schemata are sets of search points sharing some "syntactic feature"

$$\begin{aligned} s &\in \{0, 1, *\}^{\ell} \\ x &\in s, \ \textbf{iff} \ \forall i \, (x_i = s_i) \vee (s_i = *) \end{aligned}$$

For example:

"*" ⇔ "Don't care"

Schema	Members
1**0	1000 1010 1100 1110

Walsh Functions (1)

Let's define some functions for convenience...

$$\alpha(s_i) = \begin{cases} 0, & \text{if } s_i = * \\ 1, & \text{if } s_i = 0, 1 \end{cases}$$

A "0" indicates an undefined position, while a "1" indicates one that is defined.

Walsh Functions (1)

Let's define some functions for convenience...

$$\alpha(s_i) = \begin{cases} 0, & \text{if } s_i = * \\ 1, & \text{if } s_i = 0, 1 \end{cases}$$

A "0" indicates an undefined position, while a "1" indicates one that is defined.

$$j_p(s) = \sum_{i=1}^{\ell} \alpha(s_i) 2^{i-1}$$

Defines the partition number of a schema. E.g., $j_p(***) = 0$, $j_p(**f) = 1$, . . ., $j_p(fff) = 7$.

Walsh Functions (1)

Let's define some functions for convenience...

$$\alpha(s_i) = \begin{cases} 0, & \text{if } s_i = * \\ 1, & \text{if } s_i = 0, 1 \end{cases}$$

A "0" indicates an undefined position, while a "1" indicates one that is defined.

$$j_p(s) = \sum_{i=1}^{\ell} \alpha(s_i) 2^{i-1}$$

Defines the partition number of a schema. E.g., $j_p(***) = 0$, $j_p(**f) = 1$, ..., $j_p(fff) = 7$.

$$y_i = (-1)^{x_i}$$

Define auxiliary string, where $1 \mapsto -1$ and $0 \mapsto 1$. Multiplication can now act as XOR.

Walsh Functions (2)

Define ***Walsh Functions***, which provide the set of 2^ℓ monomials of aux string variables:

$$\psi_j(y) = \prod_{k=1}^{\ell} y_k^{j_k}$$

Here j is treated like a binary string, and is indexed by k.

Walsh Functions (2)

Define ***Walsh Functions***, which provide the set of 2^ℓ monomials of aux string variables:

$$\psi_j(y) = \prod_{k=1}^{\ell} y_k^{j_k}$$

Here j is treated like a binary string, and is indexed by k.

For example:

Notice how j determines which y_i values are included in the product.

Partition	$\alpha(s)$	$j(s)$	$\psi_j(y)$
$***$	000	0	1
$**f$	001	1	y_1
$*f*$	010	2	y_2
$*ff$	011	3	y_1y_2
$f**$	100	4	y_3
$f*f$	101	5	y_1y_3
$ff*$	110	6	y_2y_3
fff	111	7	$y_1y_2y_3$

Walsh Functions (3)

A brief segue (we'll come back to this later)...

- Note that we only care about values of y_k which are -1 (or when $x_k = 1$)
- In fact, we only care about the *number* of such factors included in the product
- This number is simply the number of positions k that both x and j contain a 1 $\left(x^T j\right)$
- We could re-write the Walsh Function as follows: $\psi_j(x) = (-1)^{\left(x^T j\right)}$
- Rather than see this as a set of functions that produce vectors, we could see it as a *matrix*: ψ_{xj}

Walsh Functions (4)

Things of note about Walsh Functions:

- Since $y_i \in \{-1, +1\}$, exponents > 1 are redundant
- Number in bit-reversed order (trad. in Walsh lit.)
- ψ_j defines a basis over some real vector ($\vec{w}$), just as the delta function did earlier over $\vec{f}$:

$$f(x) = \sum_{j=00\ldots0}^{11\ldots1} w_j \; \psi_j \left(y(x)\right)$$

- The ψ_j basis is orthogonal:

$$\sum_{x=00\ldots0}^{11\ldots1} \psi_i \left(y(x)\right) \; \psi_j \left(y(x)\right) = \begin{cases} 2^{\ell}, & \text{if } i = j \\ 0, & \text{if } i \neq j \end{cases}$$

Walsh Coefficients & Schema Avg (1)

- We call w_j a ***Walsh Coefficient***
- We might calculate these as follows:

$$w_j = \frac{1}{2^\ell} \sum_{x=00\ldots0}^{11\ldots1} f(x)\, \psi_j\,(y(x))$$

- However, there exists a ***Fast Walsh Transform***, similar to the Fast Fourier Transform
- Once obtained, we can use then in linear summations to produce schema averages

Walsh Coefficients & Schema Avg (2)

The relationship between the Walsh coefficients and schema averages can be obtained as follows:

$$
\begin{aligned}
f(s) &= \frac{1}{|s|} \sum_{x \in s} f(x) \\
&= \frac{1}{|s|} \sum_{x \in s} \sum_{j=00\ldots0}^{11\ldots1} w_j \, \psi_j\left(y(x)\right) \\
&= \frac{1}{|s|} \sum_{j=00\ldots0}^{11\ldots1} w_j \sum_{x \in s} \psi_j\left(y(x)\right) \\
&= \frac{1}{|s|} \sum_{j=00\ldots0}^{11\ldots1} w_j \sum_{x \in s} \quad \textstyle\prod_{i=1}^{\ell} \left(y_i(x_i)\right)^{j_i}
\end{aligned}
$$

Walsh Coefficients & Schema Avg (2)

The relationship between the Walsh coefficients and schema averages can be obtained as follows:

$$
\begin{aligned}
f(s) &= \frac{1}{|s|}\sum_{x\in s} f(x) \\
&= \frac{1}{|s|}\sum_{x\in s}\sum_{j=00\ldots0}^{11\ldots1} w_j\,\psi_j\,(y(x)) \\
&= \frac{1}{|s|}\sum_{j=00\ldots0}^{11\ldots1} w_j\,\sum_{x\in s}\psi_j\,(y(x)) \\
&= \frac{1}{|s|}\sum_{j=00\ldots0}^{11\ldots1} w_j\,\sum_{x\in s} \prod_{i=1}^{\ell}\,(y_i(x_i))^{j_i}
\end{aligned}
$$

Each term must be ± 1. The sum is bounded by $\pm|s|$.

Walsh Coefficients & Schema Avg (2)

The relationship between the Walsh coefficients and schema averages can be obtained as follows:

$$
\begin{aligned}
f(s) &= \frac{1}{|s|} \sum_{x \in s} f(x) \\
&= \frac{1}{|s|} \sum_{x \in s} \sum_{j=00\ldots0}^{11\ldots1} w_j \, \psi_j\left(y(x)\right) \\
&= \frac{1}{|s|} \sum_{j=00\ldots0}^{11\ldots1} w_j \, \sum_{x \in s} \psi_j\left(y(x)\right) \\
&= \frac{1}{|s|} \sum_{j=00\ldots0}^{11\ldots1} w_j \, \sum_{x \in s} \prod_{i=1}^{\ell} \left(y_i(x_i)\right)^{j_i}
\end{aligned}
$$

Each term must be ± 1. The sum is bounded by $\pm|s|$.

Non-zero terms always have magnitude $\pm|s|$, so we can remove $\frac{1}{|s|}$ factor.

Walsh Coefficients & Schema Avg (2)

The relationship between the Walsh coefficients and schema averages can be obtained as follows:

$$
\begin{aligned}
f(s) &= \frac{1}{|s|} \sum_{x \in s} f(x) \\
&= \frac{1}{|s|} \sum_{x \in s} \sum_{j=00\ldots0}^{11\ldots1} w_j \, \psi_j\left(y(x)\right) \\
&= \frac{1}{|s|} \sum_{j=00\ldots0}^{11\ldots1} w_j \sum_{x \in s} \psi_j\left(y(x)\right) \\
&= \frac{1}{|s|} \sum_{j=00\ldots0}^{11\ldots1} w_j \sum_{x \in s} \prod_{i=1}^{\ell} \left(y_i(x_i)\right)^{j_i}
\end{aligned}
$$

Each term must be ± 1. The sum is bounded by $\pm|s|$.

Non-zero terms always have magnitude $\pm|s|$, so we can remove $\frac{1}{|s|}$ factor.

$\therefore f(s) = \sum_j sign(s) \, w_j$

Walsh Coefficients & Schema Avg (3)

Now we consider schema averages as the partial sum of signed Walsh coefficients:

Partition	Average Fitness
$***$	w_0
$**f$	$w_0 \pm w_1$
$*f*$	$w_0 \pm w_2$
$*ff$	$w_0 \pm w_1 \pm w_2 \pm w_3$
$f**$	$w_0 \pm w_4$
$f*f$	$w_0 \pm w_1 \pm w_4 \pm w_5$
$ff*$	$w_0 \pm w_2 \pm w_4 \pm w_6$
fff	$w_0 \pm w_1 \pm w_2 \pm w_3 \pm w_4 \pm w_5 \pm w_6 \pm w_7$

For example:

$$f(*01) = w_0 - w_1 + w_2 - w_3$$

Coefficients represent the contributions linear & non-linear components of a given schema have on fitness.

Walsh Coefficients & Schema Avg (3)

Now we consider schema averages as the partial sum of signed Walsh coefficients:

Partition	Average Fitness
$***$	w_0
$**f$	$w_0 \pm w_1$
$*f*$	$w_0 \pm w_2$
$*ff$	$w_0 \pm w_1 \pm w_2 \pm w_3$
$f**$	$w_0 \pm w_4$
$f*f$	$w_0 \pm w_1 \pm w_4 \pm w_5$
$ff*$	$w_0 \pm w_2 \pm w_4 \pm w_6$
fff	$w_0 \pm w_1 \pm w_2 \pm w_3 \pm w_4 \pm w_5 \pm w_6 \pm w_7$

For example:

$$f(*01) = w_0 - w_1 + w_2 - w_3$$

Coefficients represent the contributions linear & non-linear components of a given schema have on fitness.

Walsh Coefficients & Schema Avg (3)

Now we consider schema averages as the partial sum of signed Walsh coefficients:

Partition	Average Fitness
$***$	w_0
$**f$	$w_0 \pm w_1$
$*f*$	$w_0 \pm w_2$
$*ff$	$w_0 \pm w_1 \pm w_2 \pm w_3$
$f**$	$w_0 \pm w_4$
$f*f$	$w_0 \pm w_1 \pm w_4 \pm w_5$
$ff*$	$w_0 \pm w_2 \pm w_4 \pm w_6$
fff	$w_0 \pm w_1 \pm w_2 \pm w_3 \pm w_4 \pm w_5 \pm w_6 \pm w_7$

For example:

$$f(*01) = w_0 - w_1 + w_2 - w_3$$

Coefficients represent the contributions linear & non-linear components of a given schema have on fitness.

Walsh Coefficients & Schema Avg (3)

Now we consider schema averages as the partial sum of signed Walsh coefficients:

Partition	Average Fitness
$***$	w_0
$**f$	$w_0 \pm w_1$
$*f*$	$w_0 \pm w_2$
$*ff$	$w_0 \pm w_1 \pm w_2 \pm w_3$
$f**$	$w_0 \pm w_4$
$f*f$	$w_0 \pm w_1 \pm w_4 \pm w_5$
$ff*$	$w_0 \pm w_2 \pm w_4 \pm w_6$
fff	$w_0 \pm w_1 \pm w_2 \pm w_3 \pm w_4 \pm w_5 \pm w_6 \pm w_7$

For example:

$$f(*01) = w_0 - w_1 + w_2 - w_3$$

Coefficients represent the contributions linear & non-linear components of a given schema have on fitness.

Walsh Coefficients & Schema Avg (3)

Now we consider schema averages as the partial sum of signed Walsh coefficients:

Partition	Average Fitness
$***$	w_0
$**f$	$w_0 \pm w_1$
$*f*$	$w_0 \pm w_2$
$*ff$	$w_0 \pm w_1 \pm w_2 \pm w_3$
$f**$	$w_0 \pm w_4$
$f*f$	$w_0 \pm w_1 \pm w_4 \pm w_5$
$ff*$	$w_0 \pm w_2 \pm w_4 \pm w_6$
fff	$w_0 \pm w_1 \pm w_2 \pm w_3 \pm w_4 \pm w_5 \pm w_6 \pm w_7$

For example:

$$f(*01) = w_0 - w_1 + w_2 - w_3$$

Coefficients represent the contributions linear & non-linear components of a given schema have on fitness.

Notice: Low order schema require very few Walsh coefficients

Example Fitness Function (1)

$$OneMax(x) = \sum_{i=0}^{\ell} x_i$$

x	$f(x)$	j	Part	w_j
000	0.0	0	$***$	+1.50
001	1.0	1	$**f$	-0.50
010	1.0	2	$*f*$	-0.50
011	2.0	3	$*ff$	0
100	1.0	4	$f**$	-0.50
101	2.0	5	$f*f$	0
110	2.0	6	$ff*$	0
111	3.0	7	fff	0

Example Fitness Function (1)

$$OneMax(x) = \sum_{i=0}^{\ell} x_i$$

x	$f(x)$	j	Part	w_j
000	0.0	0	$* * *$	+1.50
001	1.0	1	$* * f$	-0.50
010	1.0	2	$*f*$	-0.50
011	2.0	3	$*ff$	0
100	1.0	4	$f * *$	-0.50
101	2.0	5	$f * f$	0
110	2.0	6	$ff*$	0
111	3.0	7	fff	0

Order 1 (and 0) schema are the only contributions.

Example Fitness Function (2)

An arbitrary bitwise linear function:

$$f(x) = 10 + 5x_1 - 10x_2 + 0.1x_3$$

x	$f(x)$	j	Part	w_j
000	10.0	0	$***$	+7.55
001	15.0	1	$**f$	-2.50
010	0.0	2	$*f*$	+5.00
011	5.0	3	$*ff$	0
100	10.1	4	$f**$	-0.05
101	15.1	5	$f*f$	0
110	0.1	6	$ff*$	0
111	5.1	7	fff	0

Example Fitness Function (2)

An arbitrary bitwise linear function:

$$f(x) = 10 + 5x_1 - 10x_2 + 0.1x_3$$

x	$f(x)$	j	Part	w_j
000	10.0	0	$***$	+7.55
001	15.0	1	$**f$	-2.50
010	0.0	2	$*f*$	+5.00
011	5.0	3	$*ff$	0
100	10.1	4	$f**$	-0.05
101	15.1	5	$f*f$	0
110	0.1	6	$ff*$	0
111	5.1	7	fff	0

Again, orders 1 & 0 schema are the only contributions.

Example Fitness Function (2)

An arbitrary bitwise linear function:

$$f(x) = 10 + 5x_1 - 10x_2 + 0.1x_3$$

x	$f(x)$	j	Part	w_j
000	10.0	0	$***$	+7.55
001	15.0	1	$**f$	-2.50
010	0.0	2	$*f*$	+5.00
011	5.0	3	$*ff$	0
100	10.1	4	$f**$	-0.05
101	15.1	5	$f*f$	0
110	0.1	6	$ff*$	0
111	5.1	7	fff	0

Again, orders 1 & 0 schema are the only contributions.

This is true of all 1-separable fitness landscapes.

Example Fitness Function (3)

$$LeadingOnes(x) = \sum_{i=1}^{\ell} \prod_{j=1}^{i} x_j$$

x	$f(x)$	j	Part	w_j
000	0.0	0	$*\,*\,*$	+0.875
001	0.0	1	$*\,*\,f$	-0.125
010	0.0	2	$*f*$	-0.375
011	0.0	3	$*ff$	+0.125
100	1.0	4	$f\,*\,*$	-0.875
101	1.0	5	$f\,*\,f$	+0.125
110	2.0	6	$ff*$	+0.375
111	3.0	7	fff	-0.125

Example Fitness Function (3)

$$LeadingOnes(x) = \sum_{i=1}^{\ell} \prod_{j=1}^{i} x_j$$

Question:
Can we use lower order coeff. to approximate the opt., $x = 111$?

x	$f(x)$	j	Part	w_j
000	0.0	0	$***$	+0.875
001	0.0	1	$**f$	-0.125
010	0.0	2	$*f*$	-0.375
011	0.0	3	$*ff$	+0.125
100	1.0	4	$f**$	-0.875
101	1.0	5	$f*f$	+0.125
110	2.0	6	$ff*$	+0.375
111	3.0	7	fff	-0.125

"In some sense GAs stochastically hillclimb in the space of schemata rather than in the space of binary strings" (Rana et al., 1998)

Example Fitness Function (3)

$$LeadingOnes(x) = \sum_{i=1}^{\ell} \prod_{j=1}^{i} x_j$$

Question:
Can we use lower order coeff. to approximate the opt., $x = 111$?

x	$f(x)$	j	Part	w_j
000	0.0	0	$***$	+0.875
001	0.0	1	$**f$	-0.125
010	0.0	2	$*f*$	-0.375
011	0.0	3	$*ff$	+0.125
100	1.0	4	$f**$	-0.875
101	1.0	5	$f*f$	+0.125
110	2.0	6	$ff*$	+0.375
111	3.0	7	fff	-0.125

0^{th} order: $w_0 = 0.875$

> "In some sense GAs stochastically hillclimb in the space of schemata rather than in the space of binary strings" (Rana et al., 1998)

Example Fitness Function (3)

$$LeadingOnes(x) = \sum_{i=1}^{\ell} \prod_{j=1}^{i} x_j$$

Question:
Can we use lower order coeff. to approximate the opt., $x = 111$?

x	$f(x)$	j	Part	w_j
000	0.0	0	$* * *$	+0.875
001	0.0	1	$* * f$	-0.125
010	0.0	2	$* f *$	-0.375
011	0.0	3	$* f f$	+0.125
100	1.0	4	$f * *$	-0.875
101	1.0	5	$f * f$	+0.125
110	2.0	6	$f f *$	+0.375
111	3.0	7	$f f f$	-0.125

0^{th} order: $w_0 = 0.875$
1^{st} order: $w_0 - w_1 - w_2 - w_4 = 2.25$

"In some sense GAs stochastically hillclimb in the space of schemata rather than in the space of binary strings" (Rana et al., 1998)

Example Fitness Function (3)

$$LeadingOnes(x) = \sum_{i=1}^{\ell} \prod_{j=1}^{i} x_j$$

Question:
Can we use lower order coeff. to approximate the opt., $x = 111$?

x	$f(x)$	j	Part	w_j
000	0.0	0	$***$	+0.875
001	0.0	1	$**f$	-0.125
010	0.0	2	$*f*$	-0.375
011	0.0	3	$*ff$	+0.125
100	1.0	4	$f**$	-0.875
101	1.0	5	$f*f$	+0.125
110	2.0	6	$ff*$	+0.375
111	3.0	7	fff	-0.125

0^{th} order: $w_0 = 0.875$
1^{st} order: $w_0 - w_1 - w_2 - w_4 = 2.25$
2^{nd} order:
$w_0 - w_1 - w_2 + w_3 - w_4 + w_5 + w_6 = 2.875$

> "In some sense GAs stochastically hillclimb in the space of schemata rather than in the space of binary strings" (Rana et al., 1998)

Example Fitness Function (3)

$$LeadingOnes(x) = \sum_{i=1}^{\ell} \prod_{j=1}^{i} x_j$$

Question:
Can we use lower order coeff. to approximate the opt., $x = 111$?

x	$f(x)$	j	Part	w_j
000	0.0	0	$* * *$	+0.875
001	0.0	1	$* * f$	-0.125
010	0.0	2	$* f *$	-0.375
011	0.0	3	$* f f$	+0.125
100	1.0	4	$f * *$	-0.875
101	1.0	5	$f * f$	+0.125
110	2.0	6	$f f *$	+0.375
111	3.0	7	$f f f$	-0.125

0^{th} order: $w_0 = 0.875$
1^{st} order: $w_0 - w_1 - w_2 - w_4 = 2.25$
2^{nd} order:
$w_0 - w_1 - w_2 + w_3 - w_4 + w_5 + w_6 = 2.875$
Exact: 3.0

"In some sense GAs stochastically hillclimb in the space of schemata rather than in the space of binary strings" (Rana et al., 1998)

Example Fitness Function (3)

$$LeadingOnes(x) = \sum_{i=1}^{\ell} \prod_{j=1}^{i} x_j$$

Question:
Can we use lower order coeff. to approximate the opt., $x = 111$?

x	$f(x)$	j	Part	w_j
000	0.0	0	$***$	+0.875
001	0.0	1	$**f$	-0.125
010	0.0	2	$*f*$	-0.375
011	0.0	3	$*ff$	+0.125
100	1.0	4	$f**$	-0.875
101	1.0	5	$f*f$	+0.125
110	2.0	6	$ff*$	+0.375
111	3.0	7	fff	-0.125

0^{th} order: $w_0 = 0.875$
1^{st} order: $w_0 - w_1 - w_2 - w_4 = 2.25$
2^{nd} order:
$w_0 - w_1 - w_2 + w_3 - w_4 + w_5 + w_6 = 2.875$
Exact: 3.0

Lower order building blocks correctly predict optimum

"In some sense GAs stochastically hillclimb in the space of schemata rather than in the space of binary strings" (Rana et al., 1998)

Constructing Deception (1)

- Traditional view of GA demands that low order walsh coefficients "predict" higher order ones
- Now it is easy to see how a deceptive function can be constructed:
  - When low order estimates fail to predict the optimum
  - E.g., For two bit problem where $f(11) > f(00), f(01), f(10)$ but $f(*0) > f(*1)$ or $f(0*) > f(1*)$
  - Here $w_1 > 0$ or $w_2 > 0$ permit this

Constructing Deception (2)

Let's construct a fully deceptive, 3-bit problem:

To be deceptive, we need:

$$w_1 + w_3 > 0,\ w_2 + w_3 > 0, \text{and } w_1 + w_2 > 0$$
$$w_1 + w_5 > 0,\ w_4 + w_5 > 0, \text{and } w_1 + w_4 > 0$$
$$w_6 + w_4 > 0,\ w_6 + w_2 > 0, \text{and } w_2 + w_4 > 0$$

Constructing Deception (2)

Let's construct a fully deceptive, 3-bit problem:

To be deceptive, we need:

$$w_1 + w_3 > 0,\ \ w_2 + w_3 > 0, \text{and } w_1 + w_2 > 0$$
$$w_1 + w_5 > 0,\ \ w_4 + w_5 > 0, \text{and } w_1 + w_4 > 0$$
$$w_6 + w_4 > 0,\ \ w_6 + w_2 > 0, \text{and } w_2 + w_4 > 0$$

To preserve optimality:

$$\begin{aligned}
-(w_1 + w_2 + w_4) &> w_7 \\
w_3 + w_5 &> w_2 + w_7 \\
w_3 + w_6 &> w_1 + w_7 \\
w_5 + w_3 &> w_2 + w_4 \\
w_5 + w_6 &> w_4 + w_7 \\
w_5 + w_6 &> w_1 + w_2 \\
w_6 + w_3 &> w_1 + w_4
\end{aligned}$$

Constructing Deception (2)

Let's construct a fully deceptive, 3-bit problem:

To be deceptive, we need:

$$w_1 + w_3 > 0,\ \ w_2 + w_3 > 0, \text{and } w_1 + w_2 > 0$$
$$w_1 + w_5 > 0,\ \ w_4 + w_5 > 0, \text{and } w_1 + w_4 > 0$$
$$w_6 + w_4 > 0,\ \ w_6 + w_2 > 0, \text{and } w_2 + w_4 > 0$$

To preserve optimality:

$$\begin{aligned}
-(w_1 + w_2 + w_4) &> w_7 \\
w_3 + w_5 &> w_2 + w_7 \\
w_3 + w_6 &> w_1 + w_7 \\
w_5 + w_3 &> w_2 + w_4 \\
w_5 + w_6 &> w_4 + w_7 \\
w_5 + w_6 &> w_1 + w_2 \\
w_6 + w_3 &> w_1 + w_4
\end{aligned}$$

We can do this by enumerating the coefficients by j from 0 to 6, as long as $w_7 \leq -7$.

Constructing Deception (2)

Let's construct a fully deceptive, 3-bit problem:

To be deceptive, we need:

$$w_1 + w_3 > 0,\ \ w_2 + w_3 > 0, \text{and } w_1 + w_2 > 0$$
$$w_1 + w_5 > 0,\ \ w_4 + w_5 > 0, \text{and } w_1 + w_4 > 0$$
$$w_6 + w_4 > 0,\ \ w_6 + w_2 > 0, \text{and } w_2 + w_4 > 0$$

To preserve optimality:

$$\begin{aligned}
-(w_1 + w_2 + w_4) &> w_7 \\
w_3 + w_5 &> w_2 + w_7 \\
w_3 + w_6 &> w_1 + w_7 \\
w_5 + w_3 &> w_2 + w_4 \\
w_5 + w_6 &> w_4 + w_7 \\
w_5 + w_6 &> w_1 + w_2 \\
w_6 + w_3 &> w_1 + w_4
\end{aligned}$$

We can do this by enumerating the coefficients by j from 0 to 6, as long as $w_7 \leq -7$.

Constructing Deception (3)

A fully deceptive 3-bit fitness landscape:

x	$f(x)$	j	Part	w_j
000	13.0	0	$* * *$	0
001	11.0	1	$* * f$	+1.0
010	7.0	2	$*f*$	+2.0
011	-15.0	3	$*ff$	+3.0
100	-1.0	4	$f * *$	+4.0
101	-15.0	5	$f * f$	+5.0
110	-15.0	6	$ff*$	+6.0
111	15.0	7	fff	-8.0

Constructing Deception (3)

A fully deceptive 3-bit fitness landscape:

x	$f(x)$	j	Part	w_j
000	13.0	0	$***$	0
001	11.0	1	$**f$	+1.0
010	7.0	2	$*f*$	+2.0
011	-15.0	3	$*ff$	+3.0
100	-1.0	4	$f**$	+4.0
101	-15.0	5	$f*f$	+5.0
110	-15.0	6	$ff*$	+6.0
111	15.0	7	fff	-8.0

0^{th} order:	0
1^{st} order:	0 - 7 = -7
2^{nd} order:	-7 + 14 = 7
Exact:	7 - (-8) = 15

Constructing Deception (3)

A fully deceptive 3-bit fitness landscape:

x	$f(x)$	j	Part	w_j
000	13.0	0	$***$	0
001	11.0	1	$**f$	+1.0
010	7.0	2	$*f*$	+2.0
011	-15.0	3	$*ff$	+3.0
100	-1.0	4	$f**$	+4.0
101	-15.0	5	$f*f$	+5.0
110	-15.0	6	$ff*$	+6.0
111	15.0	7	fff	-8.0

0^{th} order:	0
1^{st} order:	0 - 7 = -7
2^{nd} order:	-7 + 14 = 7
Exact:	7 - (-8) = 15

$\boxed{\text{The lower order blocks \textit{do not} correctly predict the optimum}}$

Operator-Adjusted Fitness

- Traditional schema theorem:
  $m(s, t+1) \geq m(s, t) \frac{f(s)}{\bar{f}} \left[1 - p_c \frac{\delta(s)}{\ell - 1} - p_m o(s) \right]$
- Can think in terms of "operator-adjusted" fitness:
  $m(s, t+1) \geq m(s, t) \frac{f'(s)}{\bar{f}}$
- Can formulate operator-adjusted Walsh coefficients, and obtain $f'(s)$ this way, as well
  $w'_j = w_j \left[1 - p_c \frac{\delta(j)}{\ell - 1} - 2 p_m o(j) \right]$
- Can compute f' in terms of w', as we did for f and w

Defining Deception

(Denote the true optimum as f^*)

- ***Near Optimal Set***: $N = \{x : f^* - f(x) \leq \epsilon\}$
- ***Op-Adj. Near Optimal Set***: $N' = \{x : f'^* - f'(x) \leq \epsilon'\}$, where $\epsilon' = \frac{f'^* - w_0}{f^* - w_0}\epsilon$
- ***Statically Deceptive***: $N - N' \neq \emptyset$
- ***Statically Easy***: $N - N' = \emptyset$
- **Strictly *Statically Easy***: $N = N'$

Defining Deception

(Denote the true optimum as f^*)

- ***Near Optimal Set***: $N = \{x : f^* - f(x) \leq \epsilon\}$
- ***Op-Adj. Near Optimal Set***: $N' = \{x : f'^* - f'(x) \leq \epsilon'\}$, where $\epsilon' = \frac{f'^* - w_0}{f^* - w_0}\epsilon$
- ***Statically Deceptive***: $N - N' \neq \emptyset$
- ***Statically Easy***: $N - N' = \emptyset$
- **Strictly *Statically Easy***: $N = N'$

Idea: Will a population stay "near" the global optimum under the influence of the genetic operators once it is there?

Defining Deception

(Denote the true optimum as f^*)

- ***Near Optimal Set***: $N = \{x : f^* - f(x) \leq \epsilon\}$
- ***Op-Adj. Near Optimal Set***: $N' = \{x : f'^* - f'(x) \leq \epsilon'\}$, where $\epsilon' = \frac{f'^* - w_0}{f^* - w_0}\epsilon$
- ***Statically Deceptive***: $N - N' \neq \emptyset$
- ***Statically Easy***: $N - N' = \emptyset$
- **Strictly *Statically Easy***: $N = N'$

Idea: Will a population stay "near" the global optimum under the influence of the genetic operators once it is there?

A flaw: Goldberg calls this "convergence point" an *attractor*. In fact, it is not necessarily one. The definition of stable fixed point and attracting fixed point are not the same.

What has been learned?

- Analysis of deception

  Measures the degree of deception in terms of the *potential shift* of points in N' due to changes in f'.

- Sensitivity analysis of deception

  Measures the degree to which small changes in post-operator fitness affect the degree of deception (Goldberg, 1989b).

- Signal to noise analysis

  Measures the ratio of information provided by schema helpful for convergence, versus information which is harmful (Rudnick, 1991).

- Walsh coefficients are insufficient to infer optima

  NP hard problems exist for which all non-zero Walsh coefficients can be computed in linear time. $\therefore$ either $P = NP$ or the exact linear and non-linear interactions of a function is insufficient to infer the global optimum in polynomial time (Rana, 1998).

Criticism of Walsh Analysis

- Deception $\neq$ difficult

  There are problems that meet "deceptive" criteria that are easy for a GA, as well as the reverse (Greffenstette, 1993) .

- Walsh analysis does not necessarily agree with intuitive notions of "deception"

  Deception is not necessarily correlated with high order Walsh coefficients (Goldberg, 1990).

- Relies on hypotheses of *Schema Theory* (e.g, BBH)

  Not everyone is convinced of ST's utility or correctness in terms of dynamical prediction (Vose, 1993).

- Misses the fundamental "useful" connection between the Walsh basis and a GA

  The Walsh transform's real power lies in its ability to simplify and expose underlying properties of transformations performed by the steps in a GA generation, not in the analysis of fitness landscapes (Vose, t.r.).

Outline of Discussion

Part I:	Overview of the Walsh Transform	$\surd$
Part II:	Walsh Analysis of Fitness	$\surd$
Part III:	Walsh Analysis of Mixing Matrices	$\longleftarrow$
Part IV:	Conclusions	

Overview of the Vose SGA & Mixing (1)

- Representation of individuals are discrete, fixed-length strings using alphabets of arbitrary cardinality (we focus on binary)
- Populations are infinite in size
- Model the effects of selection and variation in a generation as a discrete time dynamical system
- Interested in analyzing the *expected* dynamical behavior in a real GA

Overview of the Vose SGA & Mixing (2)

- Population state represented as a vector of proportions of each genotype in population:
  $$\Delta^n = \{\vec{x} : x_i \in \Re,\ x_i \geq 0,\ \sum_i x_i = 1\}$$
- Dynamical map is a composition of steps in a GA generation:
  $$\mathcal{G} = \mathcal{M} \circ \mathcal{S} \circ \mathcal{F}$$
  - $\mathcal{F}$ assigns fitness, $\mathcal{F} : \Delta^n \rightarrow \Re^n$
  - $\mathcal{S}$ redistributes proportions due to selection, $\mathcal{S} : \Re^n \rightarrow \Delta^n$
  - $\mathcal{M}$ applies mutation and recombination effects, $\mathcal{M} : \Delta^n \rightarrow \Delta^n$
- Our interest is in studying mixing, so let's simplify things:
  - $\vec{x}' = \mathcal{S}\left(\mathcal{F}\left(\vec{x}\right)\right)$
  - $\vec{x}'' = \mathcal{M}\left(\vec{x}'\right)$

The Mixing Matrix

Let σ_k be the k permutation matrix and $\oplus$ mean XOR

- Define a mixing probabilities matrix $M^{(0)}$, or just M:
  $$M = Pr[\text{parent } i \times \text{parent } j \rightarrow \text{child } 0]$$
- We obtain $M^{(k)}$ generally by permuting M:
  $$M^{(k)} = M_{i\oplus k, j\oplus k}, \ \forall i, j$$
  So, $\mathcal{M}_k = (\vec{x}')^T M^{(k)} \vec{x}'$
- Equivalently, we can permute population vectors:
  $$\mathcal{M}_k = (\sigma_k \vec{x}')^T M \, (\sigma_k \vec{x}')$$
  Or, $\vec{x}'' = \sum_{i,j} x_i x_j M_{i\oplus k, j\oplus k}$

From Fourier to Walsh (1)

- We can use linear algebra methods to performing Fourier transforms
- Define a ***Fourier Matrix*** for alphabets of arbitrary cardinality, c

$$W_{ij} = \frac{1}{\sqrt{n}} e^{\frac{2\pi\sqrt{-1}\left(i^T j\right)}{c}}$$

- Now the Fourier transform is the mapping $\vec{x} \mapsto W\vec{x}^C$

(C represents complex conjugate)

- For simplicity, we write:

$$\begin{aligned} \widehat{A} &= WA^C W^C \\ \widehat{x} &= W\vec{x}^C \end{aligned}$$

From Fourier to Walsh (2)

- In the binary case ($c = 2$), we eliminate conjugation:

$$
\begin{aligned}
W_{ij} &= \frac{1}{\sqrt{n}} e^{\frac{2\pi\sqrt{-1}\left(i^T j\right)}{c}} = \frac{1}{\sqrt{n}} e^{\pi\sqrt{-1}\left(i^T j\right)} \\
e^{z\sqrt{-1}} &= \cos(z) + \sqrt{-1}\sin(z) \quad \text{but here} \quad i^T j \quad \text{must be a whole number} \\
&\therefore \quad \sqrt{-1}\sin\left(\pi\left(i^T j\right)\right) = 0 \quad \text{and} \quad \cos\left(\pi\left(i^T j\right)\right) = \pm 1 \\
W_{ij} &= \frac{1}{\sqrt{n}} (-1)^{\left(i^T j\right)} \\
\hat{x} &= \frac{1}{\sqrt{n}} W' \vec{x}
\end{aligned}
$$

- From **Part II** we have:

$$
\begin{aligned}
w_j &= \frac{1}{2^\ell} \sum_x f(x)\, \psi_j\left(y(x)\right) = \frac{1}{n} \sum_x f(x)\, (-1)^{\left(x^T j\right)} = \frac{1}{n} \sum_x f(x) \psi_{xj} \\
\vec{w} &= \frac{1}{n} W' \vec{f}
\end{aligned}
$$

From Fourier to Walsh (2)

- In the binary case ($c = 2$), we eliminate conjugation:

$$
\begin{aligned}
W_{ij} &= \frac{1}{\sqrt{n}} e^{\frac{2\pi\sqrt{-1}\left(i^T j\right)}{c}} = \frac{1}{\sqrt{n}} e^{\pi\sqrt{-1}\left(i^T j\right)} \\
e^{z\sqrt{-1}} &= \cos(z) + \sqrt{-1}\sin(z) \quad \text{but here} \quad i^T j \quad \text{must be a whole number} \\
\therefore &\quad \sqrt{-1}\sin\left(\pi\left(i^T j\right)\right) = 0 \quad \text{and} \quad \cos\left(\pi\left(i^T j\right)\right) = \pm 1 \\
W_{ij} &= \frac{1}{\sqrt{n}} (-1)^{\left(i^T j\right)} \\
\hat{x} &= \frac{1}{\sqrt{n}} W' \vec{x}
\end{aligned}
$$

In fact, the Walsh transform *is* the Fourier transform, when $c = 2$

- From **Part II** we have:

$$
\begin{aligned}
w_j &= \frac{1}{2^\ell} \sum_x f(x)\ \psi_j\left(y(x)\right) = \frac{1}{n} \sum_x f(x)\,(-1)^{\left(x^T j\right)} = \frac{1}{n} \sum_x f(x) \psi_{xj} \\
\vec{w} &= \frac{1}{n} W' \vec{f}
\end{aligned}
$$

Fun with Matrices

- Define the ***twist*** A^* of a $n \times n$ matrix A by $(A^*)_{i,j} = A_{j \oplus i, -i}$
- Define the ***conjugate transpose*** as the transpose of the complex conjugate of a matrix, denoted A^H
- $\{H, \wedge, *\}$ are interrelated operators. For example:

$$\begin{aligned} \widehat{A}^H &= \widehat{A^H} \\ \left((A^H)^*\right)^H &= (A^*)^* = \widehat{\left(\widehat{A}\right)}^* \\ (A^H)^H &= \widehat{\widehat{A}} = \left((A^*)^*\right)^* = \text{identity} \end{aligned}$$

- **The point:** complicated sequences of these operations can be simplified

Applying the Walsh Transform to $\mathcal{M}$

- A mixing matrix is dense under positive mutation, but has a sparse Fourier transform
- If mutation is zero, $M = \widehat{M}$
- $\widehat{M^*}$ is lower triangular
- If mutation is zero, M^* is upper triangular

Applying the Walsh Transform to $\mathcal{M}$

- A mixing matrix is dense under positive mutation, but has a sparse Fourier transform
- If mutation is zero, $M = \widehat{M}$
- $\widehat{M^*}$ is lower triangular
- If mutation is zero, M^* is upper triangular

Why do we care?

Applying the Walsh Transform to $\mathcal{M}$

- A mixing matrix is dense under positive mutation, but has a sparse Fourier transform
- If mutation is zero, $M = \widehat{M}$
- $\widehat{M^*}$ is lower triangular
- If mutation is zero, M^* is upper triangular

Why do we care?

Because these are ways to simplify M for the general case, such that more complicated analysis may be more tractable.

What has been learned?

- We can use the twist to more easily obtain the differential of mixing:
  $$d\mathcal{M}_x = 2\sum_u \sigma_u^T M^* \sigma_u x_u$$
- Some mathematical properties can be elicited from transformed mixing matrix:
  - Access to the *spectrum* of M obtained through M^*
  - Types of invariances under mixing exposed by Walsh transform
  - If mutation is positive, largest eigenvalue is 2 and all other eigenvalues are *inside* the unit disk
- Efficiency improvement in calculating infinite population model from $O\left(c^{3\ell}\right)$ to $O\left(c^{\ell lg3}\right)$
- Walsh provides a way to elicit model of inverse GA
- YADGE - Yet Another Derivation of Geiringer's Equation

Outline of Discussion

Part I:	Overview of the Walsh Transform	$\surd$
Part II:	Walsh Analysis of Fitness	$\surd$
Part III:	Walsh Analysis of Mixing Matrices	$\surd$
Part IV:	Conclusions	$\longleftarrow$

What is “Walsh Analysis”?

- Analysis (of a GA) using Walsh Transform
- Analysis from perspective of the Walsh basis
- It is really just a different *viewpoint*
  - Might facilitate analysis by changing the viewpoint s.t. our intuitional ideas are exposed for deeper exploration (e.g., Goldberg)
  - Might facilitate analysis by changing the viewpoint s.t. certain types of mathematical derivations become more tractable (e.g., Vose)

What is “Walsh Analysis”?

- Analysis (of a GA) using Walsh Transform
- Analysis from perspective of the Walsh basis
- It is really just a different *viewpoint*
  - Might facilitate analysis by changing the viewpoint s.t. our intuitional ideas are exposed for deeper exploration (e.g., Goldberg)
  - Might facilitate analysis by changing the viewpoint s.t. certain types of mathematical derivations become more tractable (e.g., Vose)

> Walsh Analysis is a tool to be used in conjunction with other methods, like a pair of work goggles.

Is Walsh Analysis Useful?

- Not a fair question...depends on the context of the analysis being done
  - Is analysis of schemata and building blocks helpful? Then perhaps Walsh Analysis is helpful for studying schema theory.
  - Is understanding the properties of a dynamical systems model of a GA helpful? Then perhaps Walsh Analysis is helpful for uncovering such properties.
- There may very well be other uses of this "lens" in other contexts
- Seems powerful, but is limited by limitations of existing theory which uses it

References

Bethke, A. Genetic Algorithms as Function Optimizers. Doctoral thesis, University of Michigan. 1980

Goldberg, D. Genetic Algorithms in Search, Optimization and Machine Learning. 1989

Goldberg, D. Genetic Algorithms and Wlahs Functions: Part I, A Gentle Introduction. Compplex Systems 3. 1989

Goldberg, D. Genetic Algorithms and Wlahs Functions: Part II, Deception and Its Analysis. Compplex Systems 3. 1989

Rana, S. et al. A tractable Walsh Analysis of SAT and its Implications for Genetic Algorithms. In Proceedings from the 1998 AAAI. 1998

Rudnick, M. and Goldberg, D. Signal, Noise, and Genetic Algorithms. Technical Report. 1991

Vose, M. and Wright, A. The Simple Genetic Algorithm and the Walsh Transform: part I, Theory. Technical Report (ECJ in press). 1998

Vose, M. and Wright, A. The Simple Genetic Algorithm and the Walsh Transform: part II, The Inverse. Technical Report (ECJ in press). 1998

Vose, M. The Simple Genetic Algorithm: Foundations and Theory. 1999