Jangeun Kim 20236471

Real-Coded Genetic Algorithms and Interval-Schemata

Contents

- 1. Introduction
- 2. Interval-schemata and crossover
- 3. Failure modes of an IPGA
- 4. Empirical comparisons
- 5. Crossover versus mutation
- 6. Conclusion

1.Introduction

1.Introduction

- Main objective

  ✓ In this paper, we take on the task of giving a theoretical defense of real-coded GAs.

- Primary motivation for real-coded GA

  ✓ First, real-coding of the genes eliminates the worry that there is adequate precision so that good values are representable in the search spaceV

  ☞ does not distinguish real-coded GAs from binary coded GAs.

  ✓ Second, the range of a parameter does not have to be a power of two.

  ☞ assumed that GAs that we will be comparing will use representations that have the same range and precision for any given function.

  ✓ Third, GAs operating on real-coded genes have the ability to exploit the gradualness of functions of continuous variables

  ☞ concentrate on the third feature

2. Interval-schemata and crossover

2. Interval-schemata and crossover

- Crossover operator

✓ We use a crossover operator that is a generalization of Radcliffe's which we call blend crossover (BLX-α).

☞ Radcliffe's flat crossover chooses parameters for an offspring by uniformly picking parameter values between (inclusively) the two parents parameter values

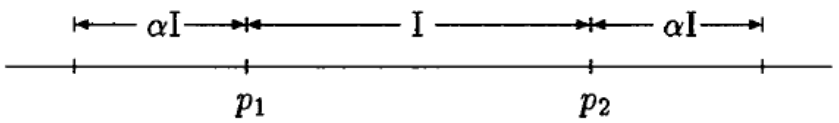


Figure 1: BLX-α

➢ For example,

- **BLX-0.5** picks parameter values from points that lie on an interval that extends $0.5I$ on either side of the interval I between the parents.(= extrema used by Wright)

- **BLX-0.0** is equivalent to Radcliffe's flat crossover.

2. Interval-schemata and crossover

- Interval-schemata

✓ What all these crossover operators have in common is that they exploit the parameter intervals determined by the parents rather than the patterns of symbols they share.

➢ suggest that the relevant concept is an **Interval-schema.**

☞ Holland's language of schemata is too restrictive for analyzing real-coded GA

➢ The number of interval-schemata that can be defined over this range of integers is: $\sum_{i=1}^{n} i = \frac{n(n-1)}{2}$

☞ Let $n = 2^L$ be the size of the range for integers that could be coded as L-bit strings

- Parameter value is a member of at least n and up to a maximum of $\lfloor (n+1)^2/4 \rfloor$ interval-schemata
- a value of k for the parameter that ranges from [0,n-1] is a member of (k+l)(n - k) interval-schemata
- two points at positions k1 and k2, k1 < k2 have (k1+ l)(n - k2) interval-schemata

☞ example

- 36 interval-schemata can be defined for a parameter whose range is [0,7].
- There are two interval-schemata of length 7, [0,6] and [1,7].
- short interval-schemata, [0,0], [1,1], etc.

2. Interval-schemata and crossover

- Interval-schemata

✓ The way an interval-processing GA (IPGA) processes interval-schemata is analogous to the way a symbol-processing GA (SPGA) processes symbol-schemata.

- ➢ it is important to note that long interval-schemata correspond roughly to low order symbol-schemata
- ➢ Both are characterized by not being very specific
- ➢ As search progresses a SPGA will progressively focus its search on higher order schemata whereas an IPGA will progressively focus on shorter interval schemata

☞ In the former case, the SPGA has narrowed the search down to certain partitions,
☞ whereas in the latter case the IPGA has narrowed the search to certain contiguous regions.

- ❖ The interval-schemata that are being searched are those bounded by the parameter extrema contained in the population.
- ❖ As these values narrow, the search becomes more and more focused, taking its samples from a smaller and smaller region of the parameter range.

3. Failure modes of an IPGA

3. Failure modes of an IPGA

- Failure to propagate good schemata

  ✓ There are a number of situations where an IPGA will have difficulty propagating good schemata, but it is instructive to consider an extreme case—a needle on a plateau.

  ☞ there is only a single value in the interval that is good, and all other values are equally bad.

  ➢ The successful algorithm requires a crossover operator that has a fairly high likelihood of passing on to the offspring those genes that are by chance the optimum allele.

  ☞ 2X has this property, since it has a relatively high probability in a many-gene problem of passing on any single gene intact.

  ☞ UX will also be more successful than BLX-0.0 at propagating optimal values when surrounded by a plateau.

3. Failure modes of an IPGA

- Failure to propagate good schemata

  ✓ If we look at the extreme case where an optimum value is crossed over with its complement, the cases appear the same.

  ➢ Example.
  Suppose the optimum lies at one of the extrema of a parameter that ranges over 2^L values

  ☞ It is coded as the integer 0 for an IPGA and a string of L zeros for a SPGA with binary coding.

  ☞ Then if the parameter is crossed over with its complement to produce a single child, there is only a $1/\ 2^L$ probability of the allele surviving in the child.

  * i.e., 2^L-1 in the case of the IPGA and L ones in the case of a SPGA

  ❖ this is not the typical situation!

3. Failure modes of an IPGA

- Failure to propagate good schemata

  - Example.
    Suppose the optimum lies at one of the extrema of a parameter that ranges over 2^L values

    - The important thing to note is that if there is no structure around the optimum, then the mate is likely to be a random individual in this range.

      - In the case of BLX-0.0 the expected value of a randomly generated gene will differ from the optimum by one half the range, and in the case of UX one half the bits.

        - **In other words**,
          the probability of propagating the optimum when mated with a randomly chosen individual is $2 * 1/ 2^L =$ **$1/ 2^{L-1}$ for BLX-0.0**, whereas it is **$1/ 2^{L/2}$ for UX**.

✓ the probability of propagating the optimum doubles for BLX to $1/ 2^{L-2}$

* Condition : the optimum lies in the center rather than an extrema

* but for intervals coded with more than four bits, UX will still have a higher likelihood of propagating the optimum than BLX-0.0.

3. Failure modes of an IPGA

- Premature convergence

✓ The strength of BLX-0.0 is that it produces its samples in the contiguous regions defined by the points contained in the population.

☞ This means that BLX-0.0 is less likely to prematurely converge to the values that correspond to the lower order bits. (We will make an important qualification to this below.)

☞ 2X, on the other hand, is much more likely to prematurely converge on the lower order bits.

✓ BLX-0.0 is good at testing small variations of contiguous chunks.

☞ whereas 2X is good at preserving contiguous chunks of the chromosome intact,

✓ BLX-0.0 has no positional bias.

☞ Unlike 2X, UX has no positional bias.
☞ It will be better at searching the lower order bits than 2X, but not as good as BLX-0.0.

❖ BLX-0.0 pays a price for this ability to exploit local information.

3. Failure modes of an IPGA

- Premature convergence

❖ BLX-0.0 pays a price for this ability to exploit local information.

- The number of interval schemata being searched by a GA using **BLX** is **limited by the maximum and minimum values** of the parameters represented in the population.

  ☞ Just as 2X or UX cannot introduce new alleles, BLX-0.0 cannot extend the interval ranges.

- If **the range of the parameters** (or cardinality of the alphabet) **is large relative to the population size**, then the algorithm is quite likely to start its search **without some values represented.**

  ☞ This is a fatal weakness for an IPGA

  e.g. if the optimal point is at one of the extrema of the interval, for a crossover operator bounded by the two points determined by the parents (as in the case of BLX-0.0) will **never be able to find the optimum unless it is enveloped by the original population**.

  ☞ generally,
  unless the extrema in the initial population envelop the optimal point, it cannot be reached via BLX-0.0

3. Failure modes of an IPGA

- Premature convergence

❖ BLX-0.0 pays a price for this ability to exploit local information.

- If the range of the parameters (or cardinality of the alphabet) is large relative to the population size, then the algorithm is quite likely to start its search without some values represented.

  ☞ This is a fatal weakness for an IPGA

  ☞ generally,
  unless the extrema in the initial population envelop the optimal point, it cannot be reached via BLX-0.0

  ⇩ How to overcome this problem?

➢ by letting the range from which an offspring is chosen extend on either side of the interval defined by the parents' parameter values (i.e., let $\alpha > 0$).

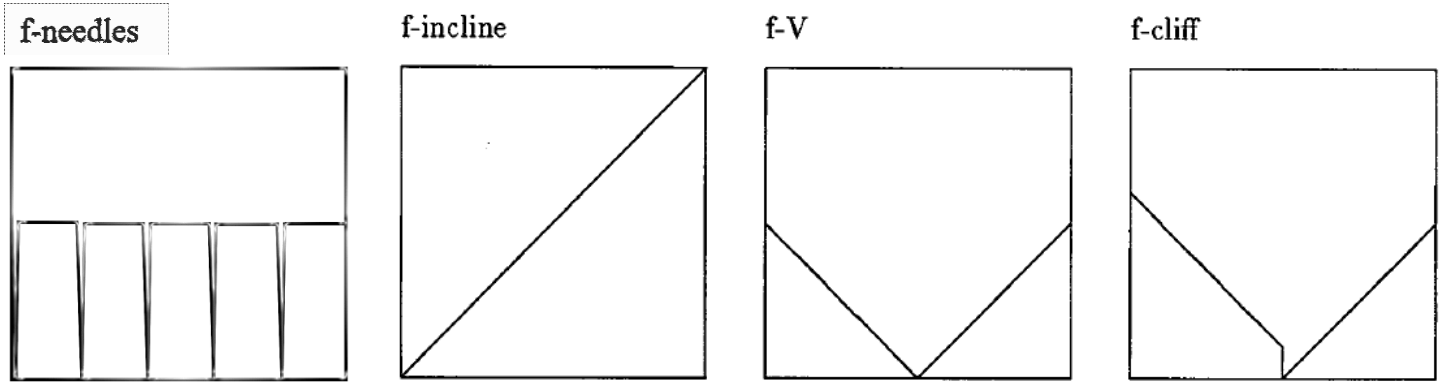
  ☞ the absence of selection pressure all values for **a < 0.5** will exhibit a tendency to population **convergence toward allele** values in the **center of their ranges**.

  ☞ **$\alpha = 0.5$** does the probability that an offspring will lie outside its parents become **equal to the probability** that it will **lie between its parents.**

4. Empirical comparisons

4. Empirical comparisons

- Failure mode tests
  - ✓ Test function
    - ➢ four test fuction : **f-needles**, **f-incline**, **f-V** and **f-cliff**



☞ **f-needles** consists of five needles on five plateaus:

* f-needles, was devised to test the hypothesis that BLX- α would have difficulty in certain circumstances propagating good schemata,

☞ **f-incline**, is a simple incline problem with the minimum (the optimum) at one extreme

☞ **f-V**, is a double incline or V function with the minimum at the center

☞ **f-cliff**, is similar to f-V except that the left incline has been raised so that there is a cliff on one side of the minimum

* above three functions were devised to test the hypothesis concerning premature convergence

* For these three functions, x ranges from 0 to 2^{30} - 1.

4. Empirical comparisons

- Failure mode tests
  - ✓ Test condition
    - ➢ tested four crossover operators: BLX-0.0, BLX-0.5, 2X, and HUX.
      - ☞ HUX is like UX, except exactly half the differing bits are swapped at random
    - ➢ tested each function using both a traditional GA and CHC
      - ☞ CHC differs from the traditional GA in several respects:
        - **(1)** **Gross generational elitist selection**: the parent and child populations are merged and the best M individuals are chosen, where M is the population size.
        - **(2)** **heterogeneous recombination (incest prevention):** only individuals who are sufficiently different (in terms of Hamming distance) are mated.
        - **(3)** **Cataclysmic mutation (restart**s): only crossover is used to produce new offspring, but when the population converges, massive mutations are applied, preserving the best individual intact, and the search is resumed using only crossover.
    - ➢ Each of the four operators produces two children per mating.
    - ➢ a population size of 50 and halted the search when either the minimum was found or the population converged (with no restarts).
    - ➢ Traditional GA used proportional selection, the elitist strategy, a population size of 50, no mutation, and 2X with a crossover rate of 1.0.

4. Empirical comparisons

- Failure mode tests

Number of times optimum found and trials to convergence								
	f-needles		f-incline		f-V		f-cliff	
	opt	trials	opt	trials	opt	trials	opt	trials
BLX-0.0	21	1597	0	1895	50	734	0	1926
BLX-0.5	20	1751	50	1418	50	968	49	1795
HUX	35	1409	47	1177	44	1153	47	1154
2X	29	831	9	1175	15	1175	8	1188
Tr-GA	11	1513	3	2111	0	1939	1	1563

- Prediction that needles-on-plateaus would be **relatively harder for BLX- α** than HUX or 2X is confirmed by BLX-0.0's and BLX-0.5's worse performance on f-needles
- as predicted, **BLX-0.0 has difficulties** on **f-incline and f-cliff**, but does **quite well on f-V** where the optimum lies in the center.
  - ☞ The good performance of BLX- 0.5 on these problems indicates that extending the interval outside the extrema determined by the parents overcomes a major shortcoming of BLX-0.0.
- Finally, the poor performance of 2X (for both a traditional GA and CHC)
  - ☞ Main failure mode for 2X on these functions is premature convergence—and in the case of the latter three functions, premature convergence on the lower order bits.

4. Empirical comparisons

- Performance tests
  - ✓ Function summary

fnc	np	bpp	len	ep	bcr	description
f1	3	10	30	N	E	parabola
f2	2	12	24	Y	H	Rosenbrock's saddle
f3	5	10	50	N	E	stair steps
f4	30	8	240	N	H	quadratic with noise
f5	2	17	34	Y	E	Shekel's foxholes
f6	2	22	44	Y	M	sine envelope sine wave
f7	2	22	44	Y	M	stretched V sine wave
f8	16	4	64	Y	H	FIR filter
f9	30	5	150	Y	H	30-city TSP, sort representation
f11	20	5	100	N	H	needle on a plateau
f12	20	5	100	N	H	deceptive
f13	10	8	80	Y	H	10 linear equations
f14	45	10	450	Y	M	dynamic control problem

fnc	function
np	number of parameters
bpp	bits per parameter
len	string length
ep	epistasis among parameters (Yes, No)
bcr	bit-climber rating (Easy, Moderate, Hard)

4. Empirical comparisons

- Performance tests
  - ✓ Test condition
    - ran CHC using BLX-0.5 and HUX on the 13 functions
    - halting each run when either the optimum was found or 50,000 evaluations had been completed
    - Unlike the failure mode tests, restarts were enabled for these runs.
    - Since f4 is noisy, it was required to be only "close" (two standard deviations) to the minimum.)
    - Test run of 50 replications

4. Empirical comparisons

- Performance tests

Mean number of trials to find optimum				
	BLX-0.5	sem	HUX	sem
f1	*874*	20	1089	25
f2	*4893*	357	9065	591
f3	2005	119	*1169*	27
f4	*933*	24	1948	97
f5	5561	588	*1396*	38
f6	14736	1998	*6496*	725
f7	3425	68	3634	291
f8	5822	522	7279	515
Mean performance				
f9	424.6	0.3	429.2	3.5
f11	1.5	0.2	*0.0*	0.0
f12	9.0	0.3	*1.2*	0.1
f13	*21.7*	2.4	61.8	11.2
f14	*16241.2*	30.1	38272.4	1039.0

- BLX-0.5 did significantly **better** than HUX for **f1, f2, f4, f13**, and **f14**,
- whereas **HUX** did significantly **better** for functions **f3**, **f5**, **f6**, **f11** and **f12**.
- For the **remaining three functions** there is **no significant difference**.

- The five problems for which BLX-0.5 is the winner are the kind of functions that one might expect BLX-0.5 to do well on.
  - ☞ They are all smooth, continuous functions.
    - **f1** and **f4** are continuous and monotonic(in Euclidean space) with independent parameters
    - **f2** is continuous and monotonic, but the discretization produces local minima in the region near the optimum.
    - **f14** is also monotonic(bit-climber can do quite well) and **f13** seems to have many local minima (tried a variety of hillclimbers on fl3, but none of them did very well)

4. Empirical comparisons

Mean number of trials to find optimum				
	BLX-0.5	sem	HUX	sem
f1	*874*	20	1089	25
f2	*4893*	357	9065	591
f3	2005	119	*1169*	27
f4	*933*	24	1948	97
f5	5561	588	*1396*	38
f6	14736	1998	*6496*	725
f7	3425	68	3634	291
f8	5822	522	7279	515
Mean performance				
f9	424.6	0.3	429.2	3.5
f11	1.5	0.2	*0.0*	0.0
f12	9.0	0.3	*1.2*	0.1
f13	*21.7*	2.4	61.8	11.2
f14	*16841.2*	30.1	38272.4	1039.0

- Performance tests

  ❖ The five cases for which BLX-0.5s poor performance

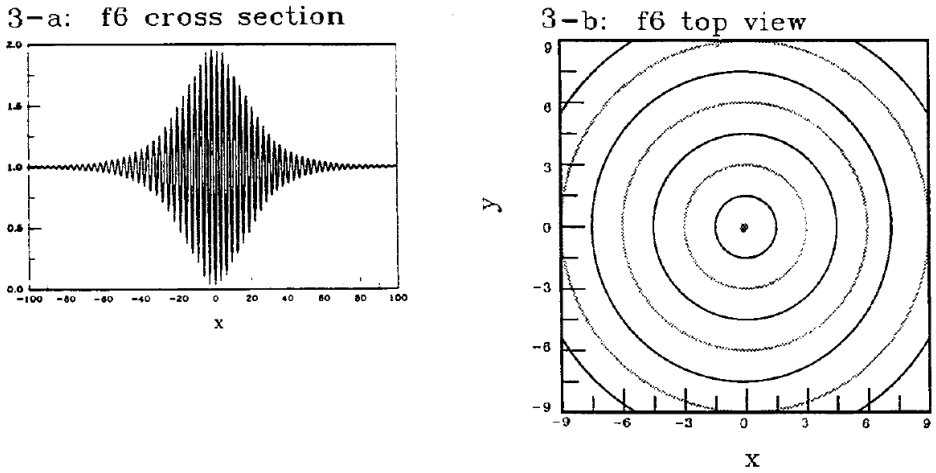
  - f11 and f12 is no continuous variables with the sort of gradualness
    - ☞ f11 consists of a 20 independent 5-bit genes, each of which is a needle on a plateau lying at one extrema
    - ☞ fl2 consists of 20 independent 5-bit genes, each of which is deceptive.
  - F3 also contains plateaus like f11
    - ☞ Each of its 5 independent 10-bit genes has more structure than those of f11 , but next to the optimum value there is a small plateau
  - f5 consists of evenly spaced wells with sloped floors sunk in a plateau. Thus, the optimum value and the 24 sub-optima are up against cliffs
  - f6 is harder to explain the BLX-0.5's poor performance!!!
    - ☞ It is a noiseless, continuous function without any plateaus or cliffs

On examination,
it turns out that **f6** illustrates **not a defect** in BLX-0.5
so much as a fortuitous advantage **presented to HUX** by the **representation chosen.**

4. Empirical comparisons

Mean number of trials to find optimum				
	BLX-0.5	sem	HUX	sem
f1	*874*	20	1089	25
f2	*4893*	357	9065	591
f3	2005	119	*1169*	27
f4	*933*	24	1948	97
f5	5561	588	*1396*	38
f6	14736	1998	*6496*	725
f7	3425	68	3634	291
f8	5822	522	7279	515
Mean performance				
f9	424.6	0.3	429.2	3.5
f11	1.5	0.2	*0.0*	0.0
f12	9.0	0.3	*1.2*	0.1
f13	*21.7*	2.4	61.8	11.2
f14	*16241.2*	30.1	38272.4	1039.0

- Performance tests
  - ❖ Examine f6 in more detail.

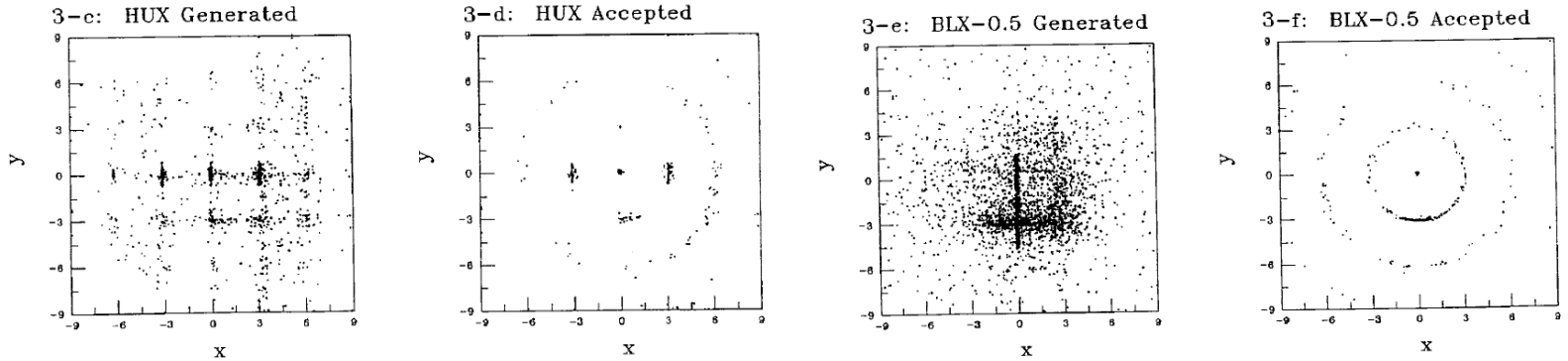


- Figure 3-a shows a **2D cross section** through the origin of f6
  - ☞ f6 is cylindrically symmetric about the z axis
- Figure 3-b shows a **small region** of f6 around the origin as seen from "above".
  - ☞ The **point** in the center is the **global optimum**, and the concentric **circles marked** with dashed lines are the regions of the **second, third**, and **fourth** best **local optima**

4. Empirical comparisons

Mean number of trials to find optimum				
	BLX-0.5	sem	HUX	sem
f1	*874*	20	1089	25
f2	*4893*	357	9065	591
f3	2005	119	*1169*	27
f4	*933*	24	1948	97
f5	5561	588	*1396*	38
f6	14736	1998	*6496*	725
f7	3425	68	3634	291
f8	5822	522	7279	515
Mean performance				
f9	424.6	0.3	429.2	3.5
f11	1.5	0.2	*0.0*	0.0
f12	9.0	0.3	*1.2*	0.1
f13	*21.7*	2.4	61.8	11.2
f14	*16241.2*	30.1	38272.4	1039.0

- Performance tests
  - ❖ Examine f6 in more detail.



☞ **Figure 3-c** plots the **points generated and evaluated** during a single run of CHC using HUX

☞ **Figure 3-d** plots the subset of points generated that are accepted into **the parent population** by replacing the worst members.

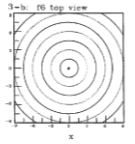
☞ **Figures 3-e** and **3-f** show **corresponding plots** for BLX-0.5.

- Figures 3-c and 3-e dramatically illustrates the difference in how schemata are sampled via a SPGA and an IPGA—patterns vs intervals.
  - ☞ see the outline of a grid-like structure filling much of Figure 3-c but not 3-e

4. Empirical comparisons

Mean number of trials to find optimum				
	BLX-0.5	sem	HUX	sem
f1	*874*	20	1089	25
f2	*4893*	357	9065	591
f3	2005	119	*1169*	27
f4	*933*	24	1948	97
f5	5561	588	*1396*	38
f6	14736	1998	*6496*	725
f7	3425	68	3634	291
f8	5822	522	7279	515
Mean performance				
f9	424.6	0.3	429.2	3.5
f11	1.5	0.2	*0.0*	0.0
f12	9.0	0.3	*1.2*	0.1
f13	*21.7*	2.4	61.8	11.2
f14	*16241.2*	30.1	38272.4	1039.0

- Performance tests
  - ❖ Examine f6 in more detail.
    - **Both algorithms** tend to get **trapped** in the **best sub-optimal region** indicated by the inner, dashed circle in Figure 3-b.



    - **Both algorithms** tend to favor points in this inner circle that **intersect the x and y axes**, although this tendency seems to be **much stronger with HUX**
    - Given that **good points** tend to **cluster** in these areas, all crossover needs to do to put a point in the central region
      - ☞ [for example] the global optimum resides, is to recombine a point in the (0, 3) with a point in the (3, 0) region
      - ➢ This would be **easy** to do with **parameter-bounded crossover**, but it turns out that in this case it **isn't** that **hard** with **uniform crossover (HUX)** either.

reason

        - ✓ the function was fortuitously discretized so that the **spacing** between the concentric circles is nearly **a power of 2.**

          * The circle marking the best sub-optima crosses the axes **65820** units from the center which is very close to 2^{16} = **65536**.

4. Empirical comparisons

Mean number of trials to find optimum				
	BLX-0.5	sem	HUX	sem
f1	*874*	20	1089	25
f2	*4893*	357	9065	591
f3	2005	119	*1169*	27
f4	*933*	24	1948	97
f5	5561	588	*1396*	38
f6	14736	1998	*6496*	725
f7	3425	68	3634	291
f8	5822	522	7279	515
Mean performance				
f9	424.6	0.3	429.2	3.5
f11	1.5	0.2	*0.0*	0.0
f12	9.0	0.3	*1.2*	0.1
f13	*21.7*	2.4	61.8	11.2
f14	*16241.2*	30.1	38272.4	1039.0

- Performance tests
  - ❖ Examine f6 in more detail.
    - ✓ the function was fortuitously discretized so that the **spacing** between the concentric circles is nearly **a power of 2.**

      * The circle marking the best sub-optima crosses the axes **65820** units from the center which is very close to 2^{16} = **65536**.

      - **Gray coded values** that are powers of **two apart** differ by **only two bits**
      - **Gray coded values** of the points where the inner sub-optimum circle crosses either of the axes will **differ** by only **two higher order bits** from the optimum point in the center.
      - **However**, Some of the **neighboring points** in this sub-optimal region will differ from the optimum by **only one bit**

      ➢ **by shifting** both the axes by an amount that is not a multiple (or a near multiple) of the distance between concentric circles, e.g., 2^{14}, the points in the best sub-optimal region will always differ from the optimum by **at least two bits.**

4. Empirical comparisons

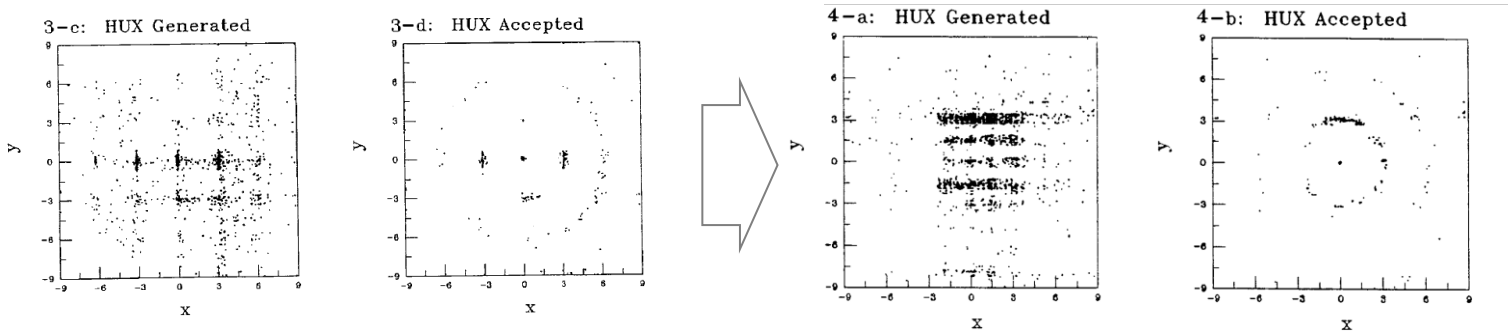
Mean number of trials to find optimum				
	BLX-0.5	sem	HUX	sem
f1	*874*	20	1089	25
f2	*4893*	357	9065	591
f3	2005	119	*1169*	27
f4	*933*	24	1948	97
f5	5561	588	*1396*	38
f6	14736	1998	*6496*	725
f7	3425	68	3634	291
f8	5822	522	7279	515
Mean performance				
f9	424.6	0.3	429.2	3.5
f11	1.5	0.2	*0.0*	0.0
f12	9.0	0.3	*1.2*	0.1
f13	*21.7*	2.4	61.8	11.2
f14	*16241.2*	30.1	38272.4	1039.0

- Performance tests
  - ❖ Examine f6 in more detail.
    - ➢ **by shifting** both the axes by an amount that is not a multiple (or a near multiple) of the distance between concentric circles, e.g., 2^{14}, the points in the best sub-optimal region will always differ from the optimum by **at least two bits.**

      ⇩

      - This makes the problem somewhat **harder for HUX**

Figure 4: Function f6-shifted (output re-centered)



☞ The grid-like pattern generated by HUX for the shifted problem is similar to that shown in **figure 3-c** except the spacing between "**lines**" is half as much as before (see Figure 4).

5. Crossover versus mutation

5. Crossover versus mutation

- ✓ Unlike the typical **mutation operator** used with a real-coded GA, BLX-0.5's "stepsize“ is self-adjusting, and is a function of the extent to which the population is converged.
  - ☞ If it is a mutation operator, it is a very special mutation operator that shares with crossover the property of increasingly focusing search.

- ✓ BLX-0.5, like all true **crossover operators**, but unlike mutation operators, including ones that are dynamically adjusted, implicitly exploits higher order correlations.
- ✓ Genes are not adjusted simply on the basis of the aggregate value of other instances of the same gene.
- ✓ Crossover implicitly takes into account the interaction among the genes when generating new instances.

6. Conclusion

6. Conclusion

- With the **new tool of interval-schemata**, the reasons behind the empirical successes reported for real-valued GAs can now **be better understood**
- Both **IPGAs** and **SPGAs** have the property of **implicit parallelism**
- They **differ** in their **biases**
- **IPGAs** exploit **local continuities**, whereas **SPGAs** exploit **discrete similarities**

Thanks