Principled Evolutionary Algorithm Design and the Kernel Trick

Fergal Lane
BDS Group
CSIS Department
University of Limerick, Ireland
fergal.lane@ul.ie

R. Muhammad Atif Azad
BDS Group
CSIS Department
University of Limerick, Ireland
atif.azad@ul.ie

Conor Ryan
BDS Group
CSIS Department
University of Limerick, Ireland
conor.ryan@ul.ie

ABSTRACT

We introduce a new approach to the principled design of *evolutionary algorithms* (EAs) based on *kernel methods*. We demonstrate how *kernel functions*, which capture useful problem domain knowledge, can be used to directly construct EA search operators. We test two kernel search operators on a suite of four challenging combinatorial optimization problem domains. These novel kernel search operators exhibit superior performance to some traditional EA search operators.

Keywords

Kernel methods; search operator design; combinatorial optimization

1. INTRODUCTION

EA configuration continues, in practice, to be a largely empirical affair. Practitioners can often be confronted with a bewildering array of possible configuration options. Typically, some subset of configuration possibilities is chosen and, then, compared and tuned using many EA runs. The broad approach of EA *principled design* is to first better understand the *problem domain* (PD) being optimized. Then, insights gained can be potentially used in a more focused, informed and efficient design process.

Several authors have previously sought to exploit problem domain structure in EA design (see [3] for a recent broad overview of work in this area). The most closely related approach to our work would be Moraglio's "geometric theory of EAs" [7] where, once one has a suitable search space metric for a problem domain, balls and line segments generated by the associated topology can be used to construct geometric mutation and crossover operators.

This paper introduces a novel approach to principled EA design. Our method begins by first finding a kernel function that matches the inherent statistical characteristics of the PD at hand. We then use this kernel function, which captures this PD knowledge, to directly construct EA search operators.

2. THE KERNEL TRICK

Algorithms such as linear regression and *principal component analyis* (PCA) principally operate using inner products in the original representation space and assume an underlying linear model for the problem/data being tackled. In principle, such linear algorithms can be extended to cope with non-linear problems/data using an explicit transform $\Phi : \mathcal{S} \rightarrow \mathcal{V}$ from the original search space $\mathcal{S}$ to some (usually higher dimensional) kernel feature space $\mathcal{V}$ where the original problem is better linearized. This approach means one has to explicitly calculate inner products for potentially huge (even infinite dimensional) and cumbersome coordinate feature vectors

The *kernel trick* [1] is where one instead uses only a kernel function $k(s,t)$ (with general form $k : \mathcal{S} \times \mathcal{S} \rightarrow \Re$) that directly gives the inner product between any two search space points s and t in some transformed kernel feature space $\mathcal{V}$. One only implicitly works in these higher dimensional spaces via cheaply-computed kernel functions. For example, a standard linear classifier that could not effectively separate data in the original space might successfully separate these in some higher dimensional feature space via a kernel; this is the basis of *Support Vector Machine* techniques in classification.

Kernel functions can also be viewed as similarity functions that give, for any two search space points, a value indicating how similar their fitness values are likely to be. Rigorous evidence-based machine learning techniques can be used to efficiently learn suitable kernel functions that capture such PD behaviour [8] (fitting a kernel function based on a sample of search space points and fitness values from several fitness functions belonging to the PD).

3. KERNEL OPERATIONS

A kernel function calculates inner products between the transformed coordinates of search space points in some kernel feature space $\mathcal{V}$. However, several other basic operations in the kernel feature space: norms, distances and squared distances, can also be expressed in terms of inner products. Table 1 lists how to calculate these using kernel functions.

However, with some basic matrix manipulation, even more complex operations in $\mathcal{V}$ (involving linear combinations, hyperplanes and spanning sets) are possible. We can construct orthonormal coordinate systems for such hyperplanes and spanning sets, and also calculate the distances of arbitrary points to these sets. Suppose $P = \{p_1, ...p_n\}, p_i \in S$ is a set of n parent points. For search operator design, two particularly useful constructs are the *parent spanning set* $\mathcal{S}_P$

Table 1: Basic Kernel Operations

Operation	Symbol	Kernel Function Implementation
Inner Product	$\langle s,t\rangle$	$k(s,t)$
Norm	$\parallel s \parallel$	$\sqrt{k(s,s)}$
Squared distance	$d^2(s,t)$	$k(s,s)+k(t,t)-2k(s,t)$
Distance	$d(s,t)$	$\sqrt{k(s,s)+k(t,t)-2k(s,t)}$

(the set of all possible linear combinations of the parent set coordinates in $\mathcal{V}$) and the parent hyperplane $\mathcal{H}_P$ (the hyperplane that intersects the n parent points). Suppose $K_{P,P}$ is the $n \times n$ matrix of kernel inner products between the parents: $(K_{P,P})_{i,j} = k(p_i, p_j)$, and $K_{P,c}$ is the n-dimensional column vector of their inner products with some arbitrary point $c \in \mathcal{S}$: $(K_{P,c})_i = k(p_i, c)$, then $[c]_E = K_{P,P}^{-\frac{1}{2}} K_{P,c}$ gives an orthonormal n-dimensional coordinate system Ψ_E for $\mathcal{S}_P$. We can use this to find the kernel distance between any two points within $\mathcal{S}_P$. The kernel distance from any point c to its closest equivalent in $\mathcal{S}_P$ can be found using the formula $d(c, \mathcal{S}_p) = \sqrt{k(c,c) - K_{P,c}^T K_{P,P}^{-1} K_{P,c}}$. Other similar formulas can be derived (as well as equivalent ones for $\mathcal{H}_P$).

4. KERNEL SEARCH OPERATORS

Using such techniques, we implemented two kernel search operators.

Our Kernel Hyperplane (KH) operator was based on the kernel distance $d(c, \mathcal{H}_P)$ between a child point c and the parent hyperplane $\mathcal{H}_P$ in $\mathcal{V}$. This had the following search operator density form:

$$\mathcal{P}_{\mathcal{KH}}(c) \propto \exp\left(-\frac{d(c,\mathcal{H}_P)}{\sigma}\right), \sigma > 0 \qquad (1)$$

The σ parameter determines how severely departures from the parent hyperplane within $\mathcal{V}$ are penalized.

Our Kernel Simplex (KS) operator was based on the kernel distance $d(c, \triangle_P)$ between a child point c and the parent simplex $\triangle_P$ in $\mathcal{V}$. This has a similar search operator density form to equation 1 (except distance to $\triangle_P$ rather than $\mathcal{H}_P$ was used). These two kernel search operators combined both crossover and mutation in a single operator. In our experiments we tested the two-parent ($n = 2$) case.

5. EXPERIMENTS AND RESULTS

Four well-known combinatorial optimization PDs with bit string search spaces were used: QUBO [4], CUBO (Cubic Unconstrained Binary Optimization) [5], NK-Landscapes [2] (with a "random neighbourhood" model without replacement), and K-Uniform MAX-SAT [6] (with $20\,n$ random clauses). Three different values for n, the bit string length, were used in simulations: $n = 25$, 50 and 100. One attraction of the PDs chosen for this suite was their actual kernel functions could be directly calculated from first principles.

We used EAs with a population size of 100 running for 100 generations. Selection was carried out using tournament selection (with size 2). We used an EA with uniform crossover and bit flip mutation (UX/BF) as a baseline comparison (a crossover rate of 0.6 and a bit flip rate of $\frac{2}{n}$ was found to gave the best all-round test suite performance). Batches of 100 runs were used to produce all experimental results given here.

Table 2: Search Operator Performances (Mean Best Run Fitness)

	Search Operator		
Problem Domain	KS	KH	UX/BF
QUBO	7.33 ± 0.03	7.28 ± 0.05	6.79 ± 0.04
CUBO	7.30 ± 0.04	7.27 ± 0.04	6.78 ± 0.04
NK (K=5)	7.16 ± 0.05	7.21 ± 0.05	6.65 ± 0.04
MAX-SAT (K=3)	6.63 ± 0.04	6.74 ± 0.04	6.36 ± 0.03
Mean Test Suite Performance	7.11 ± 0.02	7.13 ± 0.02	6.64 ± 0.02

A setting of $\sigma = \frac{1.75}{n}$ produced the best average test suite performances for the KS and KH operators. The relative performances of these search operators can be seen in Table 2. The mean performance advantage for the KS operator over UX/BF was 7% and, for the KH operator over UX/BF was 7.4% when averaged over the entire test suite (over all PDs and and problem bit string lengths).

6. CONCLUSIONS

These preliminary results show that this novel kernel-based EA principled design approach is worthy of further study. These kernel operators can be extended to exploit parent fitness information; scaling up to use multiple parents should also be straightforward. Many other kernel search operator forms are possible. Other EA components should also be amenable to kernelization, for example, diversity preservation mechanisms such as sharing. Another potentially promising avenue of research is the application of the kernel trick to *Estimation of Distribution Algorithms* (EDAs).

7. REFERENCES

[1] A. Aizerman, E. M. Braverman, and L. I. Rozoner. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and remote control*, 25:821–837, 1964.

[2] L Altenberg. NK fitness landscapes. In *Handbook of evolutionary computation*, chapter B2.7.2. Oxford University Press, 1997.

[3] Y. Borenstein and A. Moraglio. *Theory and Principled Methods for the Design of Metaheuristics*. Springer, 2014.

[4] E. Boros, P. L. Hammer, and G. Tavares. Local search heuristics for Quadratic Unconstrained Binary Optimization (QUBO). *Journal of Heuristics*, 13(2):99–132, 2007.

[5] F. Glover, J.-K. Hao, et al. Polynomial unconstrained binary optimisation – part 2. *International Journal of Metaheuristics*, 1(4):317–354, 2011.

[6] H. H. Hoos and T. Stützle. *Stochastic local search: Foundations & applications*. Elsevier, 2004.

[7] A. Moraglio and R. Poli. Topological interpretation of crossover. In *GECCO 2004*, pages 1377–1388. Springer, 2004.

[8] C. E. Rasmussen and C. K. I. Williams. Gaussian processes for machine learning. 2006. *The MIT Press*, 2006.