Towards objective measures of algorithm performance across instance space

Gadioi Alexandra

Introduction

- The paper proposes a methodology to **compare strengths and weaknesses** of different optimization algorithms **across a broader instance space**
- Issue addressed: reporting average performance of algorithms across a set of chosen instances may bias conclusions

Introduction

- Benchmark libraries help standardize the testing of algorithms, but there is a danger that algorithms are developed and **tuned to perform well on these instances**
- For many real-world or more unusual optimization problems, there is a lack of benchmark libraries, so there's a tendency for papers to be published that report algorithm performance based only on a small set of instances presented by the authors
- These papers typically are able to demonstrate that the new algorithm outperforms other approaches, and the choice of instances cannot be challenged due to lack of alternative instances

Objectives

Using the graph coloring problem as support, the methodology developed in this paper demonstrates:

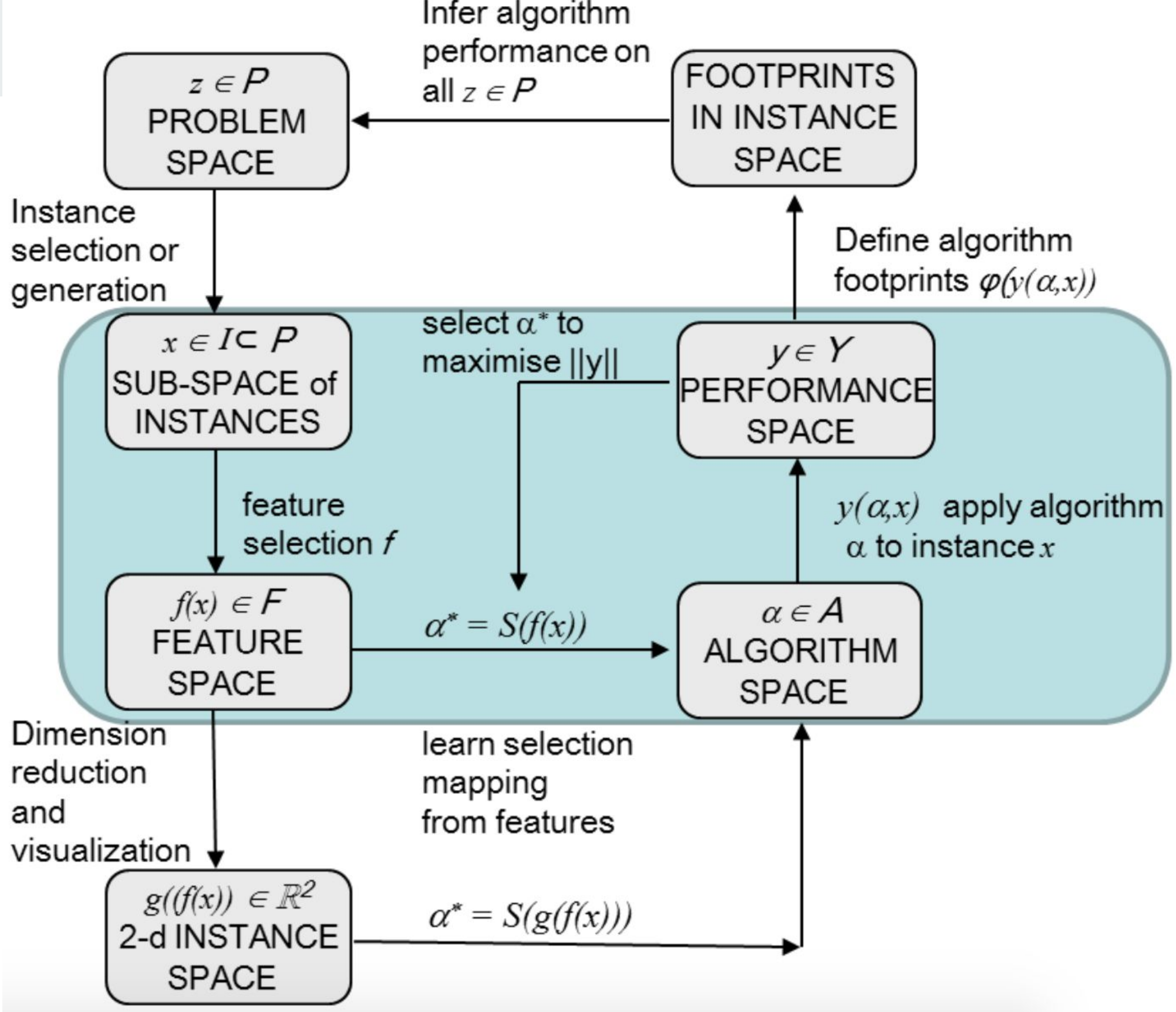
- How **pockets of the instance space** can be found where algorithm **performance varies significantly** from the average performance
- How the properties of an instance can be used to **predict algorithm performance** on previously unseen instances with high accuracy
- How the relative **strengths and weaknesses of an algorithm can be visualized** and measured objectively

Solution

Based on a framework created in 1976 (J. Rice) for Algorithm Selection Problem.

There are 4 essential components of the model:

- **P** - possibly infinitely-sized **set of instances** of a problem
- **F** - **feature space** (measurable characteristics of the instances)
- **A** - a **set of algorithms** available to solve the problem
- **Y** - performance space (mapping of **each algorithm** to a set of **performance metrics**)


Infer algorithm performance on all $z \in P$
$z \in P$ PROBLEM SPACE
FOOTPRINTS IN INSTANCE SPACE
Instance selection or generation
Define algorithm footprints $\varphi(y(\alpha,x))$
$x \in I \subset P$ SUB-SPACE of INSTANCES
select α^* to maximise $||y||$
$y \in Y$ PERFORMANCE SPACE
feature selection f
$y(\alpha,x)$ apply algorithm α to instance x
$f(x) \in F$ FEATURE SPACE
$\alpha^* = S(f(x))$
$\alpha \in A$ ALGORITHM SPACE
Dimension reduction and visualization
learn selection mapping from features
$g((f(x)) \in \mathbb{R}^2$ 2-d INSTANCE SPACE
$\alpha^* = S(g(f(x)))$

The original framework (blue box) and its extension presented in the paper

Framework details

- The collection of data describing {P,F,A,Y} is known as **meta-data**
- Rice uses only a subset of instances, the solution in paper targets all possible instances of a problem => we must consider **how the subset of instances used for meta-data are generated**
- In order to learn boundaries of algorithm performance, we must include instances that are easy and hard for each algorithm (**diversity**)
- Measuring this diversity/sufficiency will rely on **visualizing instances** in a common space, verify they are <u>spatially diverse</u> AND ensure instances are <u>discriminating of algorithm performance</u> and not equally easy or hard for all algorithms (would tell us nothing about relative strengths/weaknesses of algorithms

Methodology

Comprises 3 stages:

- Generating the **instance space**
- Algorithm **performance prediction**
- Analysis of **algorithmic power**

Generating the instance space

- Select instances
- Calculate their features
- Generate optimal subset of features
- Create a high-dimensional summary of the instances in feature-space that, when projected in **2D instance space**, achieves good separation of easy and hard instances

Feature selection (substep of instance generation)

- 2 steps:
  - Define goodness metric (metering how good is a subset of features)
  - Choose subset of instances that maximizes this metric
- Use Principal Component Analysis to create the **2D projection** based on the subset of chosen features
- Use Naive Bayes (as ML) for **discriminating between easy and hard instances**
- Use a genetic algorithm to search the space of possible subsets of m features (with the goodness measure as fitness function). Purpose: find the optimal subset of features. The one with smallest error is chosen, based on classification accuracy on an out-of-sample test set)

Visualization via dimension-reduction (substep of instance generation)

- Project the instances into 2D space
- Purpose: **visualization** of instance space and the algorithm footprint (region in instance-space where algorithm is expected to perform well)
- The 2D space should retain most of the relationships between the **instance set** and the **performance of the algorithms** on those instances

Algorithm performance prediction

- Using the 2D instance space
- The high-dimensional feature space can also be used (but the **2D instance space** has the advantage of **visualization**)
- **Assign a label to each algorithm** (for each instance) to indicate **how well the algorithm performed** (standard ML method: Naive Bayes, SVM)

Analysis of algorithmic power

- Need a user-defined description of algorithmic performance (in this paper, using binary (good/bad performance))
- Visualize the boundary in instance-space between good and bad performance for each algorithm
- Instances that lie within the boundary of good performance are deemed to be easy for that algorithm, even if those instances are not part of instance subset $I \subset P$
- We wish to measure the relative size of one algorithm’s footprint compared to another algorithm’s
- Measure used for this purpose: density of points defining the footprint. This requires a defined level of purity = percentage of instances in one algorithm’s footprint

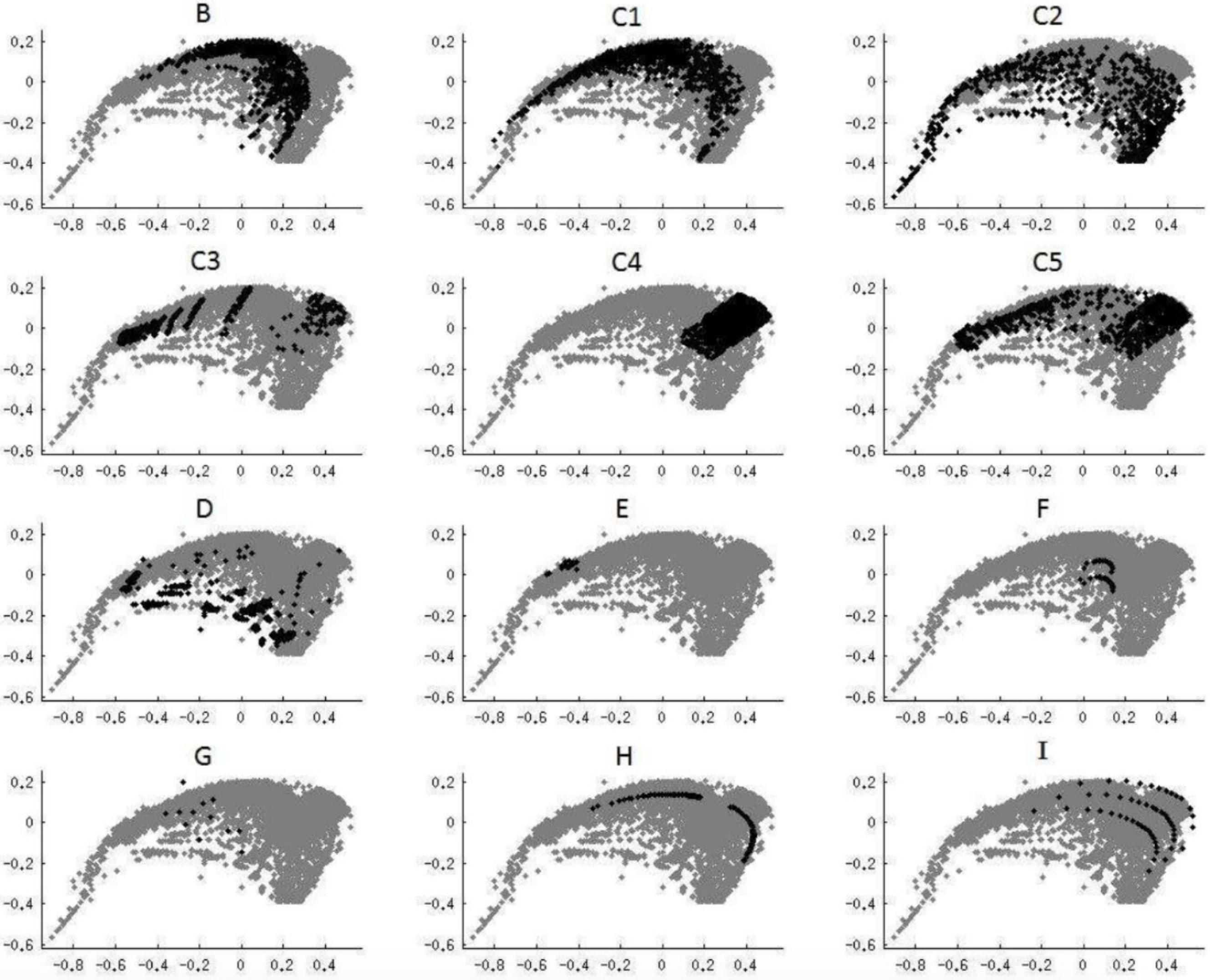
Graph coloring case study

Problem:

- A graph G=(V, E) comprises a set of vertices V and a set of edges E
- The Graph Coloring Problem is to assign colors to the vertices, **minimizing the number of colors** used, subject to the constraint that two vertices connected by an edge (**adjacent vertices) do not share the same color**
- The optimal (**minimal) number of colors** needed to solve this NP-complete problem = **chromatic number** of the graph

Instance Set	Source	Number of Instances (after outlier removal)
B	Our generator	1000 (991)
C1	Culberson [37]	1000 (1000)
C2	Culberson [37]	932 (806)
C3	Culberson [37]	1000 (1000)
C4	Culberson [37]	1000 (987)
C5	Culberson [37]	1000 (946)
D	DIMACS [38] and Networkx [39]	743 (731)
E	Social Network [19]	20 (20)
F	Sports Scheduling [19]	80 (64)
G	Timetabling [19]	13 (12)
H	Flat [19]	103 (103)
I	Random [19]	57 (52)

Table 1: Graph Coloring Instances



Each instance set shown as **black points** in the instance-space, ordered alphabetically from set B(top left) to set I(bottom right). The **grey points** define the **entire instance space**.

Algorithm	Area ($\epsilon = 0\%$)	Area ($\epsilon = 5\%$)
AntCol	19.35%	34.9%
Bktr	11.63%	14.17%
DSATUR	7.11%	12.84%
HEA	41.17%	57.14%
HillClimb	32.97%	52.08%
PartialCol	30.86%	51.84%
RandGr	0.90%	3.13%
TabuCol	36.05%	48.7%

Table 2: Relative areas of algorithm footprints for $\epsilon = 0\%$ and $\epsilon = 5\%$, expressed as a percentage of the total area of the instance space, calculated using Algorithm 1.

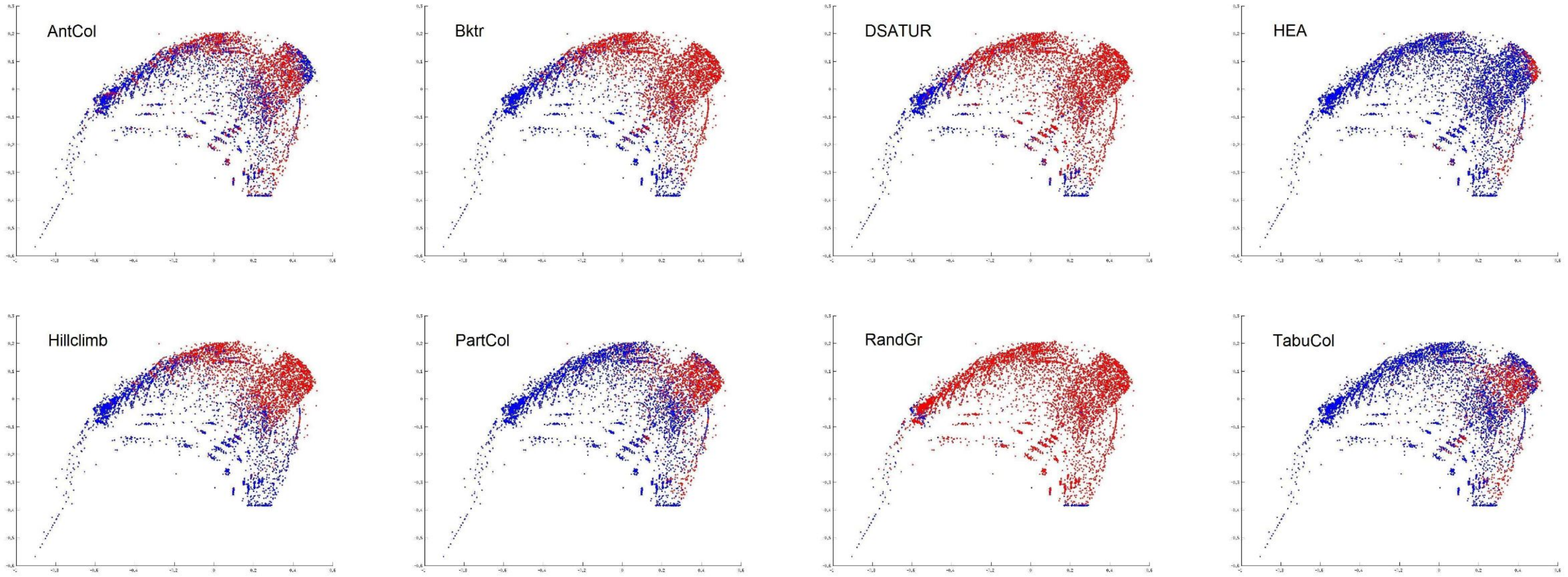


Figure 3: Algorithm Footprints showing in blue where an algorithm achieves $\epsilon - good$ performance, with $\epsilon = 0$. Red instances are not within the algorithm footprint.

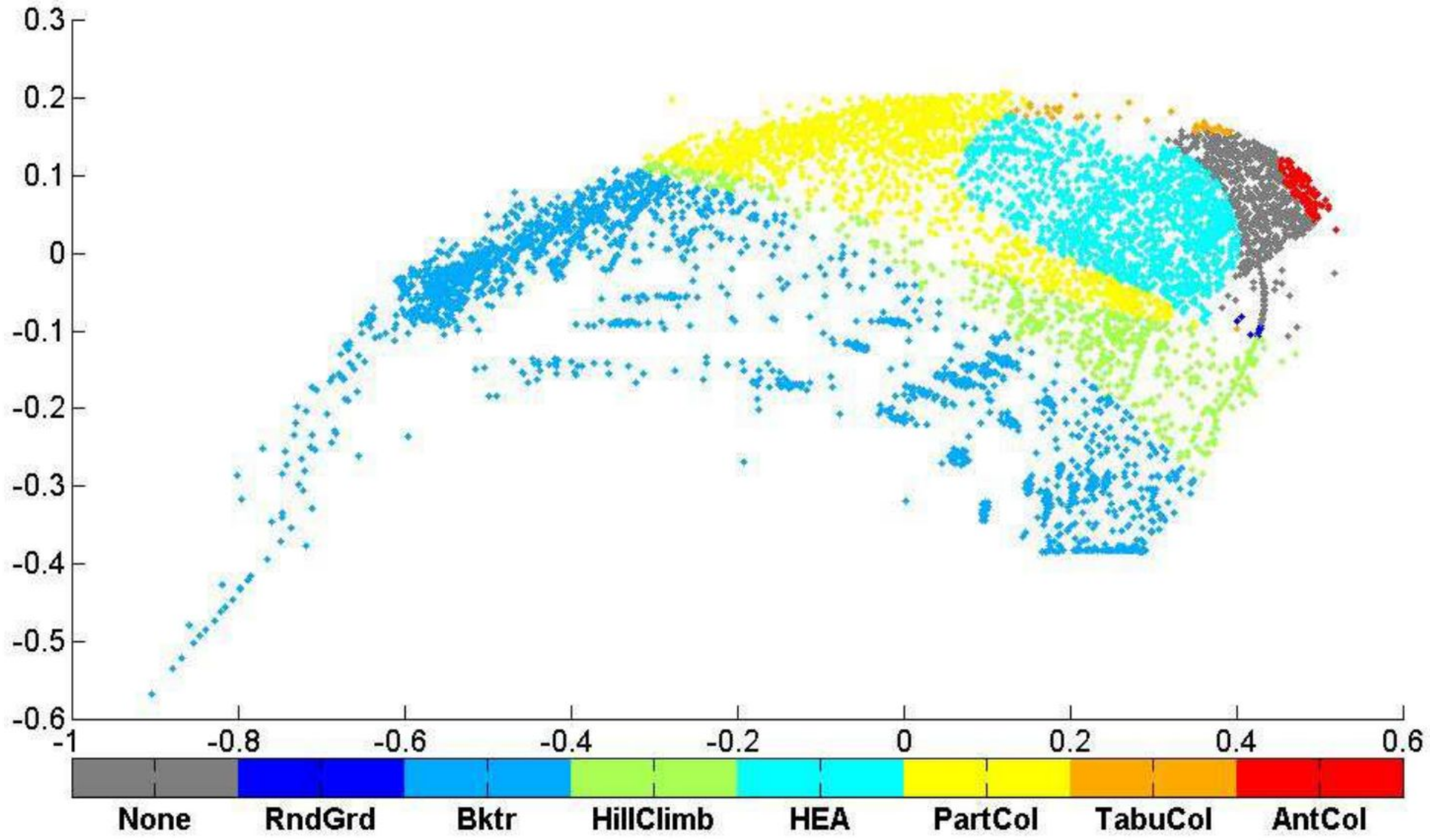


Figure 4: Machine learning (SVM) recommendations about which algorithm to use in each region.

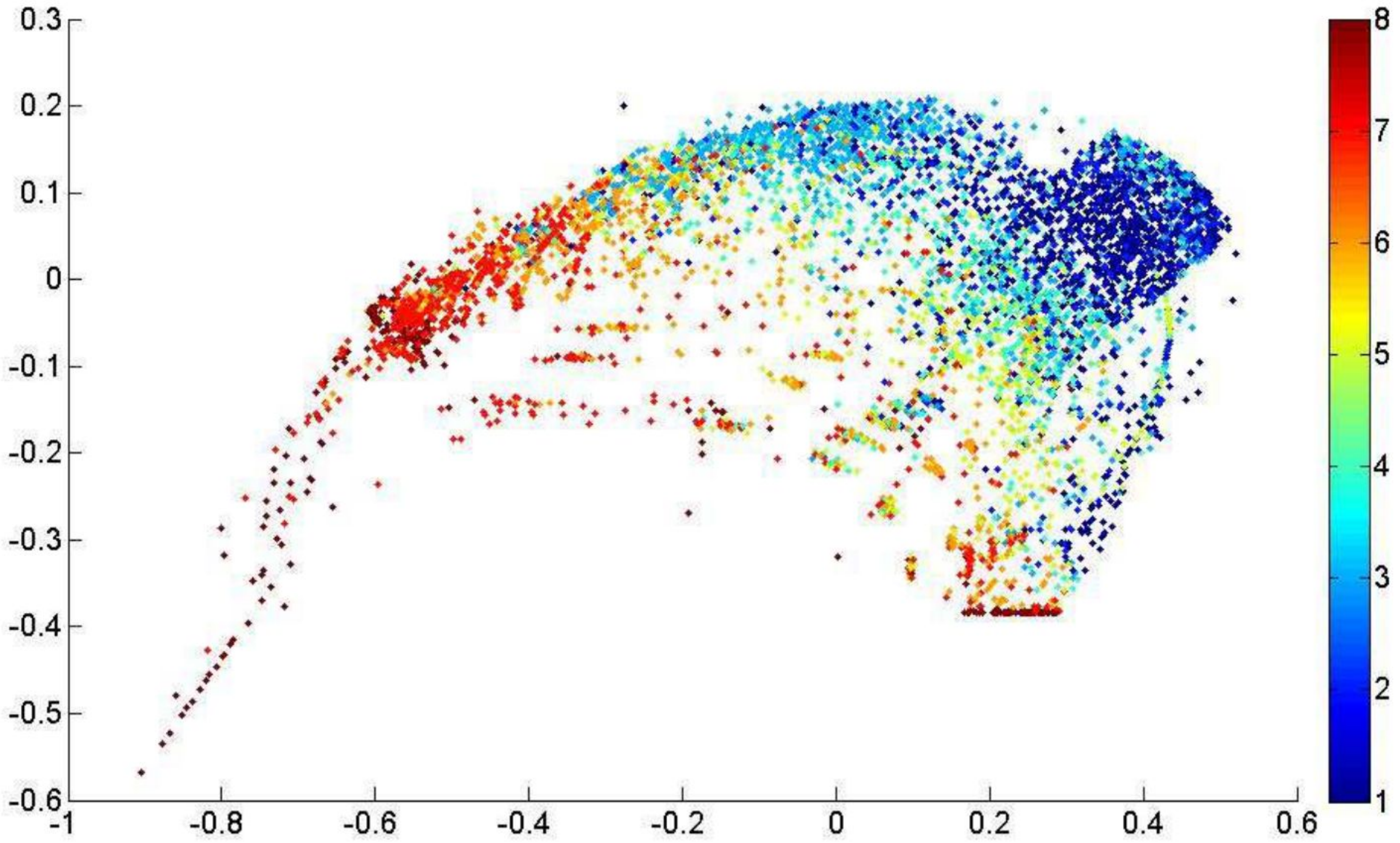


Figure 5: The number of algorithms that achieved $\epsilon - good$ performance, with $\epsilon = 0$.

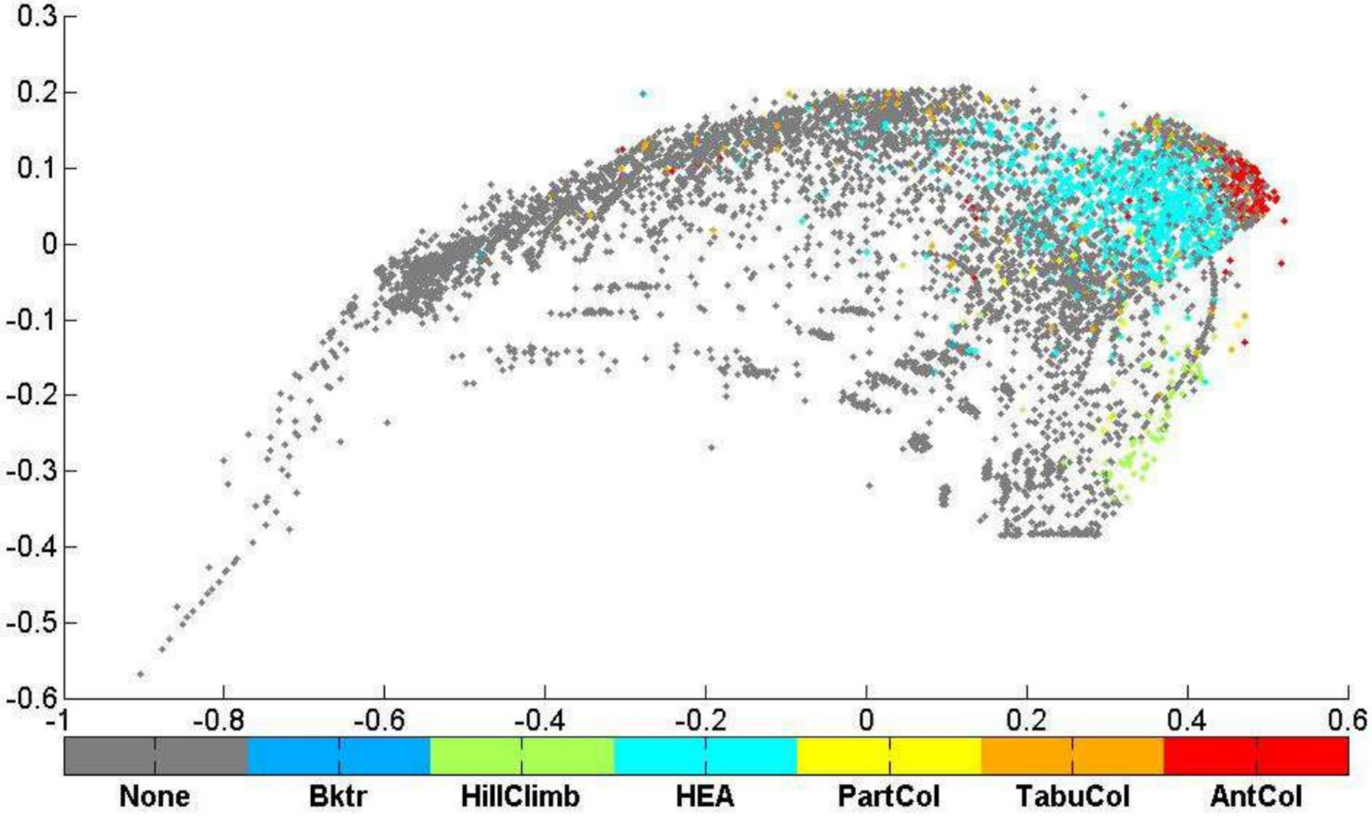


Figure 6: Instances that have a uniquely $\epsilon - good$ ($\epsilon = 0$) algorithm, color-coded by the algorithm. Grey instances are those for which there are multiple algorithms that achieved the best performance.

	$\epsilon = 0\%$		$\epsilon = 5\%$	
Algorithm	Instances (%)	Footprint Area	Instances (%)	Footprint Area
AntCol	132 (11.22%)	2.82%	5 (4.10%)	0%
Bktr	7 (0.06%)	0%	1 (0.82%)	0%
DSATUR	0 (0%)	0%	0 (0%)	0%
HEA	797 (67.77%)	15.38%	54 (44.26%)	0%
HillClimb	79 (6.72%)	2.78%	12 (9.84%)	0%
PartialCol	20 (1.70%)	0%	6 (4.92%)	0%
RandGr	0 (0%)	0%	0 (0%)	0%
TabuCol	141 (11.99%)	0.86%	44 (36.07%)	0.14%
TOTAL	1176 (17.52%)		122 (1.82%)	

Table 3: Number (Percentage) of instances, and relative area of footprints where each algorithm is uniquely $\epsilon - \mathit{good}$, for $\epsilon = 0\%$ and $\epsilon = 5\%$

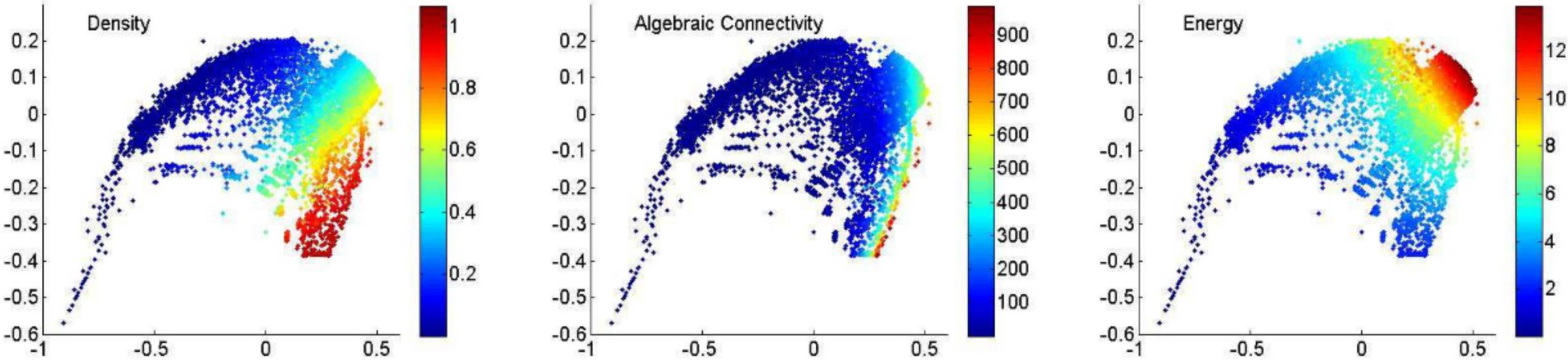


Figure 7: Distribution of the three selected features (density, algebraic connectivity and energy) across the instance space