Evolutionary n-level Hypergraph Partitioning with Adaptive Coarsening

Richard J. Preen and Jim Smith

Abstract—**Hypergraph partitioning is an NP-hard problem that occurs in many computer science applications where it is necessary to reduce large problems into a number of smaller, computationally tractable sub-problems. Current techniques use a multilevel approach wherein an initial partitioning is performed after compressing the hypergraph to a predetermined level. This level is typically chosen to produce very coarse hypergraphs in which heuristic algorithms are fast and effective. This article presents a novel memetic algorithm which remains effective on larger initial hypergraphs. This enables the exploitation of information that can be lost during coarsening and results in improved final solution quality. We use this algorithm to present an empirical analysis of the space of possible initial hypergraphs in terms of its searchability at different levels of coarsening. We find that the best results arise at coarsening levels unique to each hypergraph. Based on this, we introduce an adaptive scheme that stops coarsening when the rate of information loss in a hypergraph becomes non-linear and show that this produces further improvements. The results show that we have identified a valuable role for evolutionary algorithms within the current state-of-the-art hypergraph partitioning framework.**

Index Terms—**Evolutionary algorithms, hypergraph partitioning, memetic algorithms, multilevel algorithms.**

I. INTRODUCTION

HYPERGRAPH PARTITIONING (HGP) is an NP-hard problem [1] that occurs in many computer science applications where it is necessary to reduce large problems into a number of smaller, computationally tractable sub-problems. Common applications include very large scale integration (VLSI) design [2] and scientific computing [3].

Hypergraphs are a generalisation of graphs where each hyperedge may connect more than two vertices. Formally, a hypergraph can be defined [4], [5] as $H = \{V, \mathcal{E}, c, \omega\}$ where:

- $V = \{v_1, \ldots, v_n\}$ and $\mathcal{E} = \{e_1, \ldots, e_m\}$ are finite sets of vertices and hyperedges.
- Edges and vertices may have associated weights: $c(v)$ denotes the weight of a vertex $v \in V$ and $\omega(e)$ denotes the weight of a hyperedge $e \in \mathcal{E}$.

A hyperedge $e \in \mathcal{E}$ is said to be incident on a vertex $v \in V$ if, and only if, $v \in e$. Vertices $u, v \in V$ are said to be adjacent in a hypergraph, if, and only if, there exists a hyperedge $e \in \mathcal{E}$ such that $u \in e$ and $v \in e$. The degree of a vertex $d(v)$ is the number of distinct hyperedges in $\mathcal{E}$ that are incident on v, and the length of a hyperedge is defined as its cardinality $|e|$.

Algorithm 1: Multilevel Hypergraph Partitioning

1 **Input:** Hypergraph $H(V, \mathcal{E}, c, \omega)$, t, k, ϵ
/* coarsening */
2 **while** $|V'| > k \times t$ **do**
3     $\{a, b\} = SelectNodes(V')$
4     $V' \leftarrow Contract(V', a, b)$
5 **end**
6 $PartitionList_{V'} = InitialPartition(V', k, \epsilon)$
/* uncoarsening */
7 **while** $|V'| < |V|$ **do**
8     $c \leftarrow SelectNodeToExpand(V')$
9     $V' \leftarrow Expand(V', c)$
10     $PartitionList_{V'} \leftarrow Refine(V', PartitionList_{V'})$
11 **end**
12 **Output:** $PartitionList_V$

The k-way HGP problem is to partition the set of vertices into k approximately equal disjoint subsets whilst minimising an objective function. Typically this is the *cut-size*: the sum of the weights of those hyperedges that span different subsets. However, minimising cut-size often leads to an uneven distribution of the cut hyperedges between partitions. Alternatives are the sum of external degrees, and $(K-1)$ metric, which includes the number of subsets connected by a hyperedge [5].

Current state-of-the-art algorithms, including MLPart [6], hMetis [7], PaToH [8], Zoltan [9], Par$k$way [10], UMPa [11], and KaHyPar [12], use a multilevel approach as illustrated in Algorithm 1. The approach recursively coarsens a hypergraph by contracting a single pair of vertices at each level until $t \times k$ hypernodes remain. During coarsening, KaHyPar, hMetis, and PaToH use a greedy heavy-edge rating function in $SelectNodes()$ however more sophisticated techniques respecting the community structure have recently been explored [13]. Various methods may be used to generate the assignment of super-nodes to partitions in $InitialPartition()$. This assignment is further improved using the Fiduccia-Mattheyses [14] (FM) move-based local search algorithm. The uncoarsening phase recursively selects a node to expand (e.g., $\{a, b\} \leftarrow c$) and then uses FM to refine which partition nodes a and b are assigned. Using a larger number of levels [15] and performing repeated iterations of the entire multilevel partitioning, known as V-cycles [7], can improve the solution quality, albeit at a computational cost.

Direct k-way partitioning (Algorithm 1) has the potential advantage of allowing the search algorithm to take a global view. This can result in better solutions for large hypergraphs and tighter balance constraints [16]. However, for scalability reasons recursive bisection approaches are more widely used.

Despite their sophistication, it is notable that these approaches stop coarsening at some predefined threshold t of remaining supernodes. Most implementations, such as hMetis, PaToH, and KaHyPar, use default thresholds of $t \approx 150$, resulting in hypergraphs with around 300 vertices for initial partitioning. This value may result in fast and reasonably effective heuristic algorithms, but does not necessarily correspond to a good trade-off between scale and information content.

Karypis and Kumar [17] showed that a good partitioning of the coarsest hypergraph *generally* leads to a good partitioning of the original hypergraph. This can reduce the amount of time spent on refinement in the uncoarsening phase. However, it is important to note that the initial hypergraph partitioning with the smallest cut-size may not necessarily lead to the smallest final cut-size after refinement is performed during uncoarsening [18]. Since information may be hidden to the global optimisation algorithm during compression, the more the hypergraph is coarsened the greater this effect may be.

Many approaches have been developed to perform the initial partitioning, ranging from random assignment [6] to the use of various greedy growing techniques [8], recursive bisection [7], and evolutionary algorithms (EAs) [19]. Greedy growth algorithms quickly produce balanced partitions, but are sensitive to the initial randomly chosen vertex [8]. Since the initial partitioning usually takes place on very small hypergraphs these algorithms can be rerun multiple times. The best partitioning found is subsequently propagated for refinement during the uncoarsening phase [8].

It is difficult to generalise measures to select the optimal algorithm to use for a given problem instance, i.e., the algorithm selection problem [20]. Therefore, a portfolio approach is used in practice by PaToH, hMetis, and KaHyPar [21]. For example, PaToH uses 11 different random and greedy growth heuristic algorithms [22]. The KaHyPar 'Pool' portfolio approach to initial partitioning also uses a range of simple algorithms, including fully random, breadth-first search (BFS), label propagation, and nine variants of greedy hypergraph growing. Each algorithm is executed r number of times, then the partition with the smallest cut-size and lowest imbalance is presented for uncoarsening where it is projected back to the original hypergraph. This approach has been extensively parameter tuned [21], finding that $r = 20$ produces the overall best results at $t = 150$, with partitions that are only marginally worse than $r = 75$, yet significantly faster. Over a wide range of hypergraphs this approach has recently been shown to identify similar or better partitions in a faster time than the most popular general purpose HGP algorithms, hMetis and PaToH [12], [16], neither of which are open source.

In this article, we examine the case where there exists a large computational budget and many evaluations can be performed on less coarsened hypergraphs to identify the best final partitions, i.e., the potential for larger r and t exists. We explore the use of EAs to perform the initial partitioning within the state-of-the-art, open source (GPLv3), Karlsruhe n-level hypergraph partitioning framework, KaHyPar from https://github.com/SebastianSchlag/kahypar.

In particular, the following contributions are made:

1) We characterise the 'searchability' of the space of initial partitions at different levels of coarsening.
2) Based on that analysis, we identify a role for EAs in terms of the level of coarsening, and hence the speed vs. quality of solutions produced. We also identify some key algorithm characteristics.
3) We develop a novel memetic algorithm and demonstrate that this discovers significantly better final solutions across a range of classes of hypergraphs and across a range of different coarsening thresholds.
4) Finally, we develop an adaptive mechanism for deciding when to perform initial partitioning based on the rate of change of information content in the hypergraph as it is coarsened. We show that this also gives significant performance improvements.

In the remainder of this article, Section II discusses the related work. Section III describes the test framework, the memetic-EA initial partitioner, and comparison metrics. Section IV presents a landscape analysis with respect to EA design at different levels of coarsening. Section V presents the results of parameter sensitivity testing. Section VI introduces and presents results from a novel adaptive coarsening algorithm to identify the EA niche. Finally, Section VII summarises the conclusions.

II. RELATED WORK

Many EAs have been applied to the more well-known problem of graph partitioning; see Kim *et al.* [23] for an overview. Soper *et al.* [19] were the first to use an EA within a multilevel approach. They introduced variation operators that modify the edge weights of the graph depending on the input partitions. Subsequently presenting these to a multilevel partitioner, which uses the weights to obtain a new partition.

More recently, Benlic and Hao [24] used a memetic algorithm within a multilevel approach to solve the perfectly balanced graph partitioning problem $\epsilon = 0$. They hypothesised that a large number of vertices will always be grouped together among high quality partitions and introduced a multiparent crossover operator, with the offspring being refined with a perturbation-based tabu search algorithm.

Sanders and Schulz [25] used an EA within a multilevel approach and showed that the usage of edge weight perturbations decreases the overall quality of the underlying graph partitioner; subsequently introducing new crossover and mutation operators that avoid randomly perturbing the edge weights. Their algorithm has recently been incorporated within a faster parallelised approach [26].

In addition to performing the initial partitioning, EAs can also be used in other areas of the multilevel approach. For example, Küçükpetek *et al.* [27] used an EA to perform the coarsening phase in a multilevel graph partitioning algorithm.

Merz and Freisleben [28] showed that the fitness landscape depends on the structure of the graph and, perhaps unintuitively, that the landscape can become smoother as the average degree increases. Consequently, Pope *et al.* [29] proposed the use of genetic programming as a meta-level algorithm to select the best combination of existing algorithms for coarsening,

partitioning, and refinement, based on the characteristics of the graph being solved.

The most popular chromosome representation is group-number encoding, wherein each gene represents the partition group to assign a given vertex, i.e., there are as many genes as there are vertices $|V|$ and alleles as there are partitions. This has led to a wide variety of proposed crossover and normalisation schemes since different assignments of allele values to groups still represent the same solution. For example, Mühlenbein and Mahnig [30] used the simple normalisation technique of inverting each candidate and selecting the one with the smallest Hamming distance.

EAs have been relatively under-explored for the more general case of HGP however: there has been a small amount of prior work on VLSI circuit partitioning. For example, Schwarz and Očenášek [31] briefly studied several EAs including the Bayesian optimisation algorithm for direct (i.e., not multi-level) small VLSI partitioning. Kim *et al.* [32] explored a memetic algorithm using a modified FM for local optimisation and reported smaller bipartition cut-sizes on a number of benchmark circuits when compared with hMetis. Notably, Areibi and Yang [2] explored VLSI design via the use of memetic algorithms using FM for local optimisation within a multilevel approach and reported improvements of 35% over a simple genetic algorithm. This has since been implemented in hardware using reconfigurable computing [33]. Significantly, none of these algorithms are considered to be competitive with state-of-the-art hypergraph partitioning tools.

Recently a memetic EA has been introduced to build on the KaHyPar framework [34]. This algorithm runs a steady-state EA with a population at the original uncoarsened level. The initial population is seeded using a variant of KaHyPar. Each generation, binary tournament selection is used to choose two parents, then variation operators are applied to the fitter of those, running a number of V-cycles of coarsening–initial partitioning–uncoarsening, using different randomisation seeds. The recombination operator only runs V-cycles on the subset of original-level vertices that are in different partitions in the two parents. Two mutation operators were defined: one starting from the original level, and another which preserves more locality by skipping the coarsening phase and starting from the initial partition corresponding to the fitter parent (these are cached to save time.) To maintain diversity, a variant of restricted tournament selection is used and the authors introduce a novel distance measure that they claim is better suited to this problem domain than Hamming distance.

The work presented here and that in [34] share the idea that the memetic algorithm should work at a less coarsened level. However, there are key differences: in [34] the EA works at the wholly uncoarsened level, which can mean millions of vertices/genes. Therefore, to make the search tractable the sub-space in which search occurs (via the V-cycles) is restricted and initial partitioning run at a highly coarsened level.

III. METHODOLOGY

A. Test Framework

To ensure the comparability of results we use the KaHyPar n-level hypergraph partitioner [12], [16], [21]. This is a mature toolkit to which considerable attention has been paid to parameter tuning, so no further optimisation was applied. We also use a selection of the hypergraphs used previously for benchmarking KaHyPar, available from http://doi.org/10.5281/zenodo.30176. Specifically, we use: the 10 largest from the well-known ISPD98 VLSI circuits [35]; and 10 each randomly selected from the University of Florida sparse matrix collection (SPM) [36] (*Airfoil_2d, Reuters911, usroads, stokes128, Andrews, Baumann, HTC_336_9129, NotreDame_actors, Stanford, nasasrb*) and the 2014 international SAT competition (SAT) [37] (*gss-20-s100, MD5-28-2, ctl_4291_567_5_unsat_pre, aaai10-planning-ipc5-pathways-17-step21, slp-synthesis-aes-top29, hwmcc10-timeframe-expansion-k45-pdtvisns, dated-10-11-u, atco_enc1_opt2_05_4, UCG-15-10p1, openstacks-p30_3.085*).

Since KaHyPar is currently the best general state-of-the-art hypergraph partitioner [12], [16], and recursive bipartitioning can scale with increasing k more effectively, here we use an initial testing regime of $k = 2$ and $\epsilon = 0.1$. For benchmark comparisons, we use the KaHyPar Pool portfolio algorithm described above, and compare results at equivalent numbers of evaluations. An evaluation consists of generating an initial partitioning followed by an application of the FM algorithm. However, it should be noted that one evaluation of an algorithm in the Pool (e.g., a BFS) has a longer wall-clock time than an EA evaluation. The total partitioning times for the experiments reported here are approximately $1.9\times$ longer for the Pool when compared at the same t threshold. For $k = 2$, the $(K\text{-}1)$ and hyperedge cut-size metrics are identical [4], and so here we use this as the objective function.

B. Representation, Algorithm Operators and Parameters

We adopt a simple vertex-to-cluster encoding of the N coarsened hypernodes, and use a $(\mu+\lambda)$ EA where each subsequent generation consists of the μ fittest from the parental population and λ offspring. Each offspring is created as the product of two (independently) randomly selected parents. Uniform crossover is applied with $\mathcal{X}$ = 80% probability. Symmetry in the fitness landscape can severely obstruct the evolutionary search [38], so we apply parental alignment (normalisation) during crossover: if the Hamming distance between the parents exceeds $N/2$ then the gene values of one parent are inverted. A self-adaptive mutation scheme is then applied, setting genes to random values. Following Serpell and Smith [39], each candidate maintains its own mutation rate. This is initially inherited from the fitter of its parents, and then with $\mathcal{A}$ = 10% probability may be randomly reset to one of 10 possible values before applying mutation at the resulting rate. If an offspring has an imbalance greater than ϵ, a repair mechanism is invoked, randomly moving vertices from the largest to the smallest partition. Lamarkian evolution is performed by subsequently applying the FM local search algorithm using default [12] KaHyPar settings and the offspring acquiring any modifications. See Algorithm 2.

C. Comparison Metrics and Statistical Analysis of Results

The distribution of values observed from repeated runs was not normally distributed—especially when there is a 'hard'

Algorithm 2: Memetic EA($\mu + \lambda$) initial partitioner

1   $n = \frac{1}{|V|}$; $\mathcal{M} = [\frac{n}{100}, \frac{n}{10}, \frac{n}{5}, \frac{n}{2}, n, n, 2n, 5n, 10n, 100n]$
2   initialise parent population: $P = \{a_1 \ldots a_\mu\}$
3   **while** *evaluation budget not exhausted* **do**
     /* create offspring population */
4   **for** $i = 1$ *to* λ **do**
5   parent $p_1 = RandomSelection(P)$
6   parent $p_2 = RandomSelection(P)$
7   offspring $a_i = p_1$
8   **if** *rand() < $\mathcal{X}$* **then**
9   perform uniform crossover with normalised p_2
10   **if** $p_1.fitness < p_2.fitness$ **then**
11          $a_i.mut = p_2.mut$
12   **end**
13   **end**
14   **if** *rand() < $\mathcal{A}$* **then**
15        $a_i.mut = RandomSelection(\mathcal{M})$
16   **end**
17   **for** *each hypernode in a_i* **do**
18   **if** *drand() < $a_i.mut$* **then**
19   assign hypernode to a random partition
20   **end**
21   **end**
22   repair partition if necessary
23   apply FM local search (Lamarkian)
24   evaluate a_i
25   **end**
     /* select next parental population */
26     $P = \mu$ fittest from $P + \lambda$
27   **end**

---

lower or upper limit. We therefore apply non-parametric tests.

For each run, we recorded two values: the *initial* cut-size as the value found by a search algorithm operating at the coarsest level, and the *final* cut-size as the value at the original level, i.e., after uncoarsening has taken place. Since these values will depend on the coarsening threshold t and choice of algorithm, we denote these as cut_{alg}^t. In some cases below we also report the *best-case* cut-size: cut_{alg}^*, the value observed at whichever coarsening threshold gave the best results for a given dataset.

To measure the performance of different algorithms across the full range of thresholds, we also present the area under the curve (AUC) results, estimated from the experiments at individual thresholds using a composite Simpson's rule. When comparing methods on a single problem, we use the Wilcoxon ranked-sums test, with the null hypothesis that all observed results come from the same distribution.

To draw any firm overall conclusions about the performance of the two approaches, we follow the recommendations in [40] for comparing algorithms over multiple data sets. First, we examine the results to ensure that for each algorithm-hypergraph combination the arithmetic mean is a reliable estimate of performance, i.e., that the distribution of observations from the 20 runs is unimodal with low standard deviation. This results in a pair of values (one per algorithm) for each hypergraph, to which the Wilcoxon signed ranks test can be applied with the null hypothesis that taken across all hypergraphs there is no difference in performance.

Finally, run-times are recorded as total-wall-clock time for the whole process because the time taken in each phase is

heavily linked to the results of the previous stage.

IV. Landscape Analysis at Different Levels

One of the tenets of the multilevel approach to solving HGP is that the sheer size of the search space makes it impractical to solve at the original, uncoarsened level, and that therefore it is better to conduct the search for a good initial partitioning within a much smaller space. It has also been suggested that the graph-partitioning counterparts become easier to search as the level of coarsening increases [28]. Nevertheless, there is clearly a trade-off. It is inevitable that the coarsening process reduces the information content, so the mapping between quality of initial and final cuts becomes more noisy—especially given the greedy uncoarsening process.

To investigate the nature of the search spaces at different levels of coarsening, we used KaHyPar to generate 10000 random starting points, apply FM to each and stored these local optima. For each problem we then identified the (usually singleton) set of 'quasi-global' optima. For each local optima, we measured its Hamming distance (and that of its inverse) to each of the global optima, and recorded the smallest distance (scaled [0,1]), together with the relative cut-size, i.e., divided by the landscape's estimated global minimum. This was done at $t = 150$ and $t = 15000$ for four hypergraphs from each of ISPD98, SPM, and SAT collections.

Landscapes were examined through a combination of visual analytics (scatter and kernel-density-estimate, KDE plots) and a model of the fitness-distance correlation (FDC). The FDC model is a linear regression of local optima l in the form $cut(l_i) = m \times distance(l_i, g)$. The proportion of observed variation in relative cut-size that can be described by the model was recorded, i.e., the co-efficients of determination (R^2).

This analysis showed a significant similarity between problems, with the exception of *Stanford* where coarsening stops prematurely. Fig. 1 shows KDE plots for the two thresholds overlaid with the FDC results for two typical hypergraphs. Note the y scales were chosen to permit comparison between different thresholds and so significant numbers of local optima with high relative cut-sizes are not shown. This is why the linear regression lines lie above the main cloud of points visible at $t = 150$. The results of this analysis, and the implications for search algorithm design are:

1) On some problems the coarsening process was observed to stop prematurely, and at different values when repeated (e.g., between 34000 and 65000 hypernodes for *Stanford*). This suggests that search algorithms should be designed to cope with large search spaces.

2) The FM process greatly reduced cut-sizes and there was no correlation between the cut-sizes of solutions before and after improvement. This suggests a lack of global structure of the landscape as a whole, i.e., considering all points rather than just local optima. This indicates algorithms should incorporate local search.

3) All search landscapes contained large numbers of distinct local optima. Only a few tens of duplicates were found; more than one copy of the global optima was only found in 2 of the 24 runs, and never at $t = 15000$.
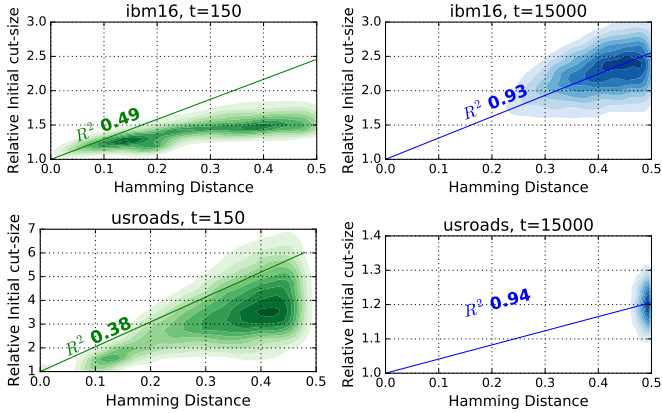
Fig. 1. The relationship between local optima initial cut-size and Hamming distance at thresholds $t = 150$ and $t = 15000$. Each graph shows a kernel density plot of the results from 10000 randomly seeded FM local searches and FDC results. y-axes are scaled to facilitate comparison between thresholds and so do not show many poor optima for $t = 150$.

It was common to see cut-sizes an order of magnitude worse than the quasi-global optimum. This suggests that it is worth devoting computational effort to finding good starting points for the search process.

4) On all landscapes there was a positive FDC, i.e., the global optimum was likely to be near other good local optimum. This mirrors previous findings on the related graph partitioning problem [28], [41]. This suggests benefits for search algorithms that can exploit this information such as population-based search with some form of recombination.

5) This effect was noticeably more present on the large landscapes ($t = 15000$). This suggests that there may be a role for population-based search in partitioning at less coarse levels than is possible with single-member search algorithms such as BFS.

6) There was almost always a 'gap' between the best solution found and next best. The lack of duplicates makes it unlikely the global optima had large basins of attraction. Given the numbers of 'good' local optima found just beyond this gap, this suggests a concentric structure. This may be because points "in the gap" are infeasible, or because the basins of attraction of the good-but-not-optimal local optima are large. Again this suggests a role for recombination, but as this has less effect as populations converge, it also suggests a changing role for mutation during search. Self-adaptation of mutation rates has often been shown successful in a wide range of domains [42] and simple approaches can be shown theoretically to be capable of overcoming both fitness and entropic barriers in combinatorial landscapes [43].

V. SENSITIVITY TO EA DESIGN CHOICES

A. Population Seeding

The landscape analysis suggests that for some hypergraphs there is good reason to devote significant effort to finding good starting points for search. To examine this hypothesis, and conversely, whether seeding is detrimental when those conditions do not apply, we exploit the portfolio of algorithms in the Pool as a selection of heuristics for quickly finding approximate solutions. To examine the performance of the EA ($\mu = 100, \lambda = 1000$) with different amounts of initial seeding, experiments were run with the EA seeded with $\mu \times s$ Pool evaluations: for example, when $s = 10$, the first 1000 evaluations are generated from the Pool before the EA begins.

In Fig. 2 the cut-sizes of the best solutions discovered are shown for the *ibm18*, *Reuters911*, *Stanford*, and *usroads* hypergraphs at coarsening threshold $t = 15000$. All results are averages of 20 runs. On both *ibm18* and *Reuters911*, the EA quickly identifies better solutions than the Pool algorithm regardless of the seeding strategy, showing that the evolutionary search is able to effectively follow a gradient in the fitness landscape. However, on *Stanford* and *usroads*, the EA without seeding ($s = 0$) performs very poorly, being an order of magnitude worse than $s = 100$ after 30000 evaluations. Given that so many local optima are present in such a fitness landscape, starting with fully random solutions ($s = 0$) or only a few good solutions ($s = 1$, $s = 10$) can cause the EA to converge prematurely. Only by starting the EA at a suitable point in the landscape, here after 10000 Pool evaluations ($s = 100$), is it able to consistently find very good solutions regardless of the effectiveness of coarsening. Further increasing the amount of seeding ($s = 200$) did not result in additional improvements. In all following experiments therefore we use $s = 100$, i.e., 10000 initial Pool evaluations.

The top-right KDE plot in Fig. 1 suggests a reason for these observations. The huge majority of local optima lie far from the global optimum and considering the high-density contours, there is little or no slope to guide the search towards the global optimum. Although there is a correlation between local optima cut-size and distance from the global optimum, this gradient only emerges when enough seeds have been considered to sample the lower-density contours of the KDE.

B. Population Size

EA sensitivity to μ and λ was explored by repeating the previous experiments across the spectrum of coarsening levels on the same 12 hypergraphs. A ratio of 1:10 was employed as this is a commonly used setting, especially with self-adaptive mutation [39]. The EA(10+100) was found to produce significantly worse final cut-sizes than EA(100+1000). However, EA(50+500) and EA(200+2000) were not significantly different than EA(100+1000). This shows that the EA is reasonably robust to these parameters and the use of 100+1000 is justified here for the use of fixed parameters. However, as shown in Table I, the optimum coarsening threshold t^* differs for each hypergraph. Therefore, adaptive population sizing schemes would further optimise wall-clock partitioning time and have been shown to increase EA performance [44].

C. Variation Operators

Further experimentation on less coarsened hypergraphs ($t = 15000$) confirmed results widely reported for graph partitioning [23] that both the use of uniform crossover and parental alignment significantly improved performance. This
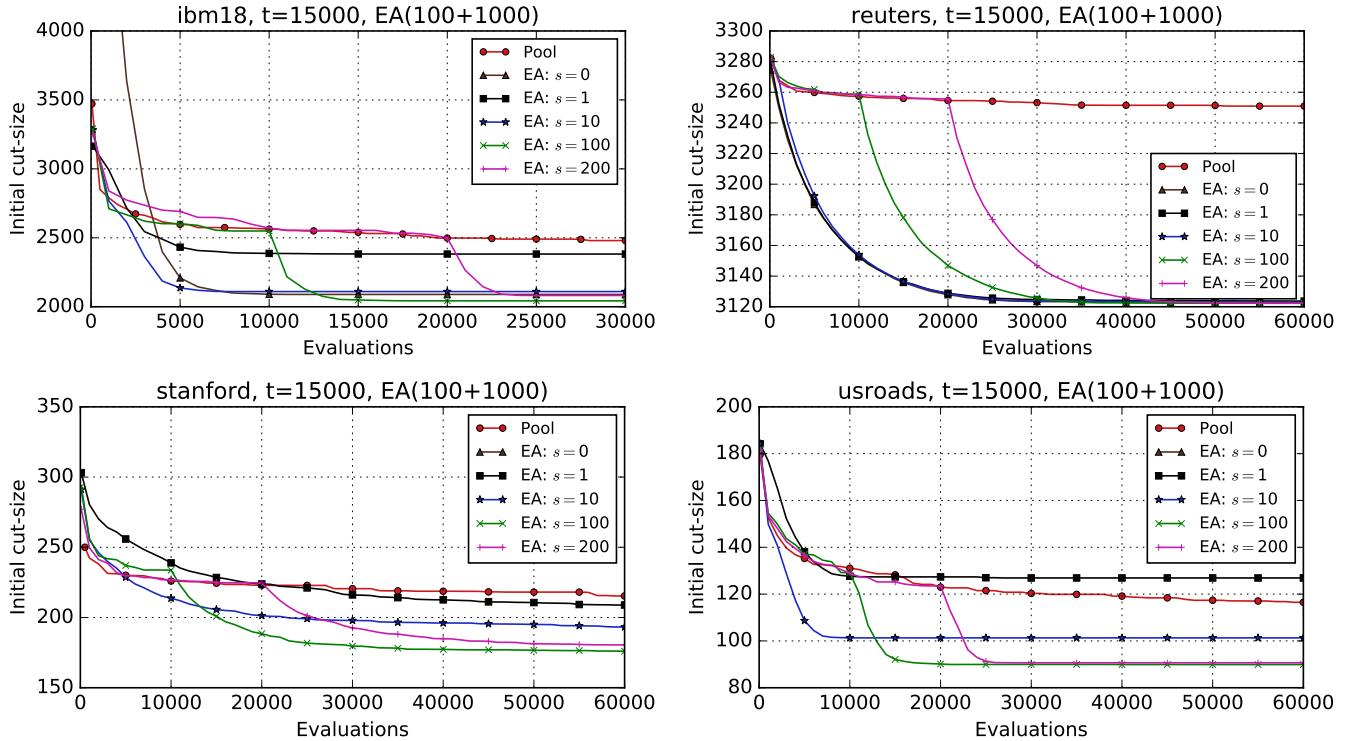
Fig. 2. The affect of population seeding on the *ibm18, Reuters911, Stanford*, and *usroads* initial partitioning. Shown are the cut-sizes of the best solutions discovered by the Pool (circle), and the EA initially seeded with $\mu \times s$ number of Pool evaluations; $\mu = 100$, $\lambda = 1000$. On the *Stanford* and *usroads* hypergraphs the EA without seeding ($s = 0$) is not observable since the cut-size values exceed the y-axis limit.

finding remained consistent even with the use of self-adaptive mutation. For example, EA(100+1000) with $\mathcal{X} = 80\%$ produced initial cut-sizes on average 30% smaller than $\mathcal{X} = 0\%$ on *ibm18* after 30000 evaluations, $p \leq 0.05$.

Estimation of distribution algorithms (EDAs) have been used to generate many state-of-the-art results by replacing recombination and mutation with a process of building and then sampling probabilistic graphical models (PGMs) of the current populations. We adapted Pelikan's implementations of the Bayesian optimisation algorithm (BOA) [45] to work within our seeding regime, and to explicitly exploit the representation's symmetry during model building. With small t no significant differences in performance were observed. However, the scalability of the model building process was an issue with large t. Runs on a MacBook Pro with a 2.8GHz 4-core Intel i7 processor with 16GB RAM were halted after 6 hours stuck in initial model building for both decision tree and graph-based variants of BOA, even after restricting the space of PGMs to bivariate models. Simplifying still further to a univariate model removed the ability to accurately capture interactions. Runs with s=100 initial seeding produced significantly larger mean initial cut-sizes after 30000 evaluations on the 4 hypergraphs in Fig. 2; 2422, 3154, 210, and 128 on *ibm18, Reuters911, Stanford* and *usroads*, respectively.

D. Search at Different Coarsening Levels

The more coarsening performed on a hypergraph before partitioning, the more information is potentially hidden from the optimisation algorithm, i.e., it must move larger blocks.

However, the less coarsening performed, the larger the search space and potentially the worse the optimisation algorithm will perform. To explore this relationship between algorithm and coarsening threshold, we examine the results of initial and final partitioning by the Pool and EA with s=100 seeding across a spectrum of coarsening levels. For each of the three classes of hypergraph, we perform experiments across the spectrum of coarsening thresholds on 4 of the 10 selected benchmark hypergraphs[1]. Additionally we ran tests at $t = 150$ and $t = 15000$ on all 30 hypergraphs. Results presented are an average of 20 runs of each algorithm run to 30000 initial partitioning evaluations at each coarsening threshold; each threshold is sampled in intervals of 250 for $t \leq 5000$, and of 5000 above that. The initial and final cut-sizes can be seen in Fig. 3.

1) Overall Performance: Using the AUC metric to compare performance across all coarsening thresholds, initial cut sizes found by the EA were smaller than those found by Pool on all 12 problems. The same is seen for final cut sizes with the exception of *Stanford*, where it should be noted that the coarsening algorithm produces hypergraphs with $|V| \geq 30000$ (200000 pins) even at $t = 150$.

2) Highly Coarsened Hypergraphs: The nature of the search landscapes for highly coarsened hypergraphs results in little difference between the algorithms. No statistically significant difference between algorithms was observed on any of the 30 benchmarks for either initial or final cut-sizes.

---

[1] *ibm 15–18; gss-20-s100, aaai, MD5-28-2*, and *slp* from the SAT collection; and SPMs *Airfoil_2d, Reuters911, Stanford*, and *usroads*.
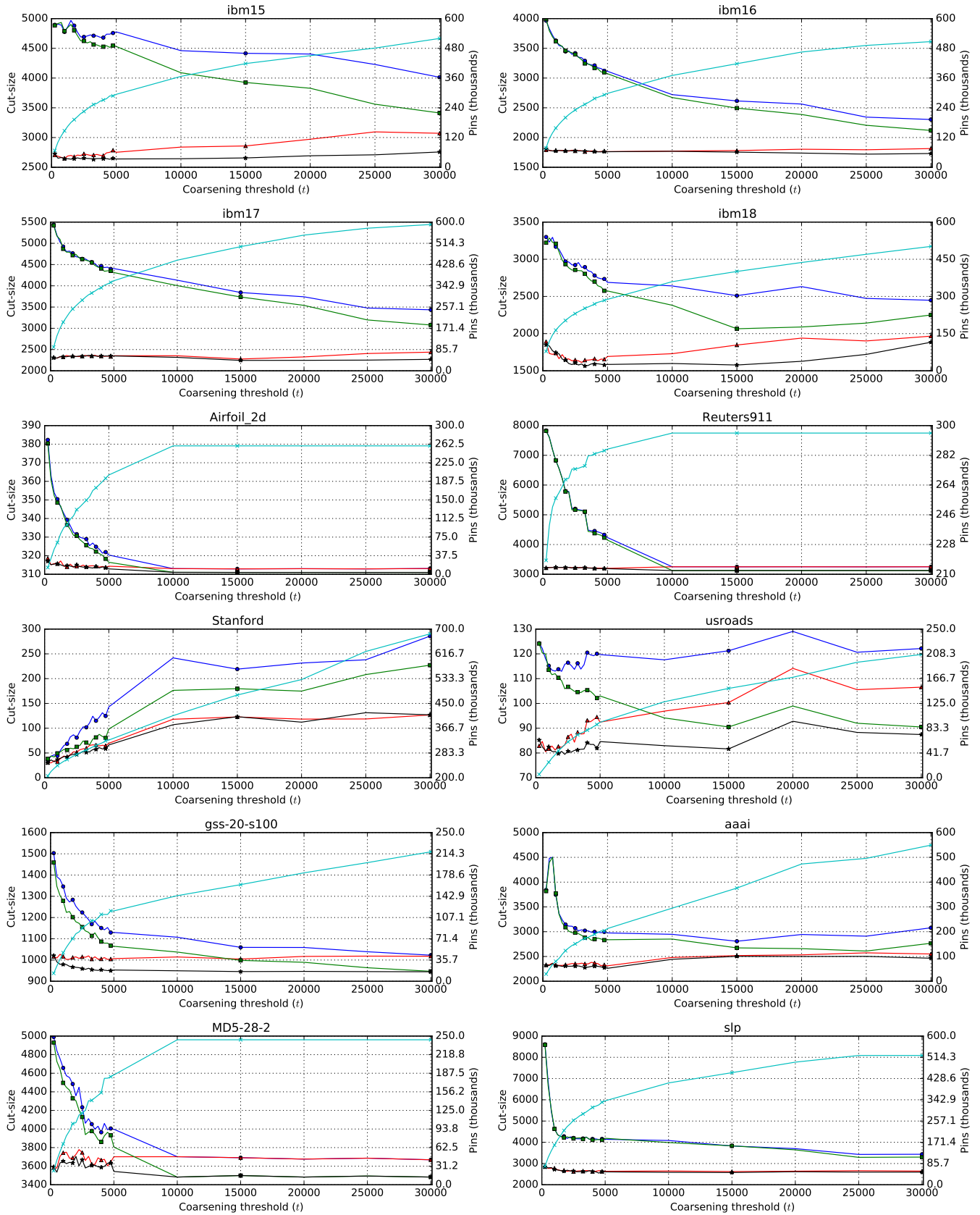
Fig. 3. Cut-sizes for the initial and final partitioning of 12 hypergraphs from the ISPD98, Univeristy of Florida Sparse Matrix Collection, and 2014 SAT competition. Shown are the results of 20 runs of the Pool and EA(100+1000) run to 30000 evaluations at each coarsening threshold, sampled in intervals of 250 for $t \leq 5000$, and in intervals of 5000 for $t > 5000$. Pool initial cut-size (circle); EA initial cut-size (square); Pool final cut-size (triangle); EA final cut-size (star); number of pins in the hypergraph (cross). For the *Airfoil_2d, Reuters911*, and *MD5-28-2* hypergraphs, $|V| \approx 15000$, therefore the affects of coarsening can only be observed at $t < 10000$.

3) Less Coarsened Hypergraphs: The difference between algorithms becomes more significant the less coarsening is performed. For example, at $t = 15000$ the EA mean best initial cut-sizes are significantly smaller than the Pool on all 10 of the ISPD98 hypergraphs (Wilcoxon rank-sum test, $p \leq 0.05$). Furthermore, these improvements in initial partitioning lead to smaller final cut-sizes. The mean and median are lower for the EA than the Pool algorithm on all 10 of the ISPD98 hypergraphs; but not significantly different at the 95% confidence interval on *ibm10* and *ibm11*. On *ibm18*, the EA mean inital and final cut-size were 20% and 16% smaller than the Pool.

Similar improvements to initial partitioning are found by the EA on the SPM hypergraphs. For example, with $t = 15000$, the EA mean initial cut-sizes on 8 of the 10 SPM hypergraphs are significantly smaller than the Pool (Wilcoxon rank-sum test, $p \leq 0.05$); no significant difference was observed on the *nasarb* and *Andrews* hypergraphs. Interestingly, despite the improvement in initial partitioning, this only resulted in significant differences in final cut-sizes on the *Airfoil_2d, Reuters911*, and *usroads* hypergraphs, where the EA resulted in improvements to mean final cut-size of 0.7%, 4%, and 15% respectively. At this t setting, no coarsening is performed on either the *Airfoil_2d* or *Reuters911* hypergraphs and therefore the cut-sizes are entirely a result of the memetic EA.

For SAT hypergraphs at $t = 15000$, both the mean EA initial and final cut-size is significantly smaller than the Pool on 6 of the hypergraphs ($p \leq 0.05$), with no significant difference on the other 4, again showing that the EA performs a more effective search on larger hypergraphs.

Performing Wilcoxon signed-ranks tests of the initial partitionings across all runs on the 10 ISPD98 hypergraphs confirms that the EA has a significantly lower cut-size than the Pool at $t = 15000$ ($p \leq 0.05$). Moreover, this also translates to significant improvements in the final partitioning ($p \leq 0.05$). Similar results were found when repeating the class tests for the 10 SPM hypergraphs and the 10 SAT hypergraphs.

4) Optimum Coarsened Hypergraphs: Table I shows the smallest (average) final cut-sizes discovered by the Pool and EA across all coarsening thresholds on the 4 hypergraphs from each benchmark set. This shows that when the optimum coarsening threshold for each algorithm-problem combination is known, the smallest final cut-size discovered by the EA is less than the Pool algorithm on all 4 of the largest ISPD98 hypergraphs. On the SAT hypergraphs, the best EA final cut-sizes are on average smaller by 5.8% on *gss-20*, 2.2% on *aaai10*, 2.75% on *MD5-28-2*, and 2.6% on *slp-synthesis*. These improvements are statistically significant for all but *ibm15* and *Stanford*. The improvements were achieved by the EA carrying out a more effective search at the same or higher coarsening threshold than the Pool and therefore able to take advantage of any additional information in the larger initial hypergraph.

Also shown in Table I is the average total EA partitioning time, $time^*_{EA}$, relative to that taken by the Pool, $time^*_{Pool}$. As can be seen, the EA is faster on 7 of the 12 hypergraphs despite operating on a similar or larger initial hypergraph.

5) Summary:
- The results for all 30 hypergraphs at the coarsest level (t=150) show no significant difference between algo-

TABLE I
THE SMALLEST (AVERAGE) EA AND POOL FINAL CUT-SIZES ON FOUR HYPERGRAPHS FROM EACH OF THE BENCHMARK SETS AND THE RELATED COARSENING THRESHOLDS. CUT-SIZE HIGHLIGHTED IN BOLD FACE WHERE IT IS SIGNIFICANTLY DIFFERENT, $p \leq 0.05$.

Hypergraph	t^*_{Pool}	t^*_{EA}	cut^*_{Pool}	cut^*_{EA}	$\frac{time^*_{EA}}{time^*_{Pool}}$
ibm15	1000	3250	2649	2632	2.69
ibm16	3250	25000	1762	**1720**	3.15
ibm17	15000	15000	2276	**2244**	0.74
ibm18	3000	3250	1612	**1564**	0.57
Airfoil_2d	15000	15000	312	**311**	0.66
Reuters911	5000	10000	3199	**3125**	0.60
Stanford	500	250	30	29	0.40
usroads	750	2250	80	**79**	1.87
aaai10-planning	5000	5000	2312	**2261**	0.65
gss-20-s100	1250	30000	1002	**944**	9.67
MD5-28-2	500	10000	3580	**3483**	6.41
slp-synthesis	2500	4500	2618	**2549**	0.96

rithms.
- However, with larger initial hypergraphs (t=15000), the EA significantly outperforms the Pool ($p \leq 0.05$).
- Furthermore, the wall-clock time of the Pool algorithm was significantly higher than the EA's ($p \leq 0.05$).

Moreover, results confirm our hypothesis that if initial partitioning is done on large hypergraphs, the picture changes dramatically. Taken as a whole, for the 12 instances where the spectrum of coarsening thresholds was explored:
- the EA significantly outperforms the Pool algorithm over all coarsening thresholds (AUC metric).
- The final cut-sizes of the EA at t^* are significantly smaller for all 12 hypergraphs than the Pool algorithm at the default t=150.
- Taking the optimum threshold for each algorithm-problem combination, and comparing the best-case cut-sizes across the 12 problems, the EA results are significantly better than the Pool algorithm ($p \leq 0.05$).

VI. ADAPTIVE COARSENING TO IDENTIFY THE EA NICHE

The less coarsening is performed, the more information may be available to the initial partitioning algorithm to potentially achieve higher quality partitions. This is particularly evident in a number of the hypergraphs in Fig. 3 by observing the final cut-sizes where $t < 5000$; see, for example, *ibm18*. However, for each algorithm there exists a point at which further increases in the size of the search space result in declining performance; for example, see the algorithm cut-sizes on the *ibm18* hypergraph where $t > 15000$ in Fig. 3. Simply selecting a fixed larger t does not help since the 'optimal' threshold is clearly hypergraph-dependent..

From Fig. 3 it can be seen that the sum of the number of vertices in each hyperedge, $|pins|$, initially declines relatively linearly with the number of hypernodes before reaching a point of exponential decay. This suggests that for each hypergraph there may exist a tipping point at the balance between maximal information content and maximal hypergraph compression, akin to 'knee-points' in Pareto fronts. We therefore propose an adaptive coarsening scheme that halts hypernode contraction in response to the changing characteristics of the hypergraph.

Algorithm 3: Adaptive coarsening stopping criteria

```
 1  R: regression buffer of length t_n
 2  while |V| > t × k do
 3      for each hypernode do
 4          select contraction partner
 5          perform contraction
 6          if |V| < t_max × k then
 7              if t_s hypernodes coarsened since last update then
 8                  update R with |pins|
 9                  r ← coefficient of linear regression on R
10                  if r² < t_r then
11                      stop coarsening
12                  end
13              end
14          end
15          if |V| < t × k then
16              stop coarsening
17          end
18      end
19  end
```

A. Algorithm

We perform a linear piecewise approximation of the curve based on a sliding window of observations, and seek to identify the knee-point at which the linear approximation is least representative of the curve. Coarsening occurs as normal until there are fewer than $t_{max} \times k$ hypernodes; here $t_{max} = 15000$. Thereafter, a linear regression is performed on $|pins|$, sampled after every t_s hypernodes have been contracted, and calculated on the most recent t_n samples. Coarsening is terminated and initial hypergraph partitioning performed as usual when the correlation coefficient $R^2 < t_r$ or the original $t = 150$ threshold reached. See Algorithm 3.

A grid search of these parameters was performed to minimise the final EA(100+1000) cut-sizes on the 12 hypergraphs for which partitioning was previously performed across the range of coarsening thresholds and the best performing parameters $t_s = 50$, $t_n = 100$ and $t_r = 0.99$ were identified.

B. Results

Results show that over a wide range of different hypergraphs this simple adaptive threshold can identify better places to stop coarsening, although with some large variations:

- Across all 30 hypergraphs there was an overall reduction in the mean final cut-size of 1.6% ($p \leq 0.05$) compared with the results achieved at $t=150$; and a 1.25% reduction ($p > 0.05$) compared with results at $t=15000$.
- The mean final cut-size is smaller on 22 of the 30 hypergraphs when using the adaptive threshold compared with the EA at $t=150$. This difference is statistically significant on 6 of the 10 ISPD98 hypergraphs, 2 of the 10 SPM hypergraphs (*Reuters911* and *usroads*) and 2 of the 10 SAT hypergraphs (*gss-20-s100* and *UCG-15-10p1*). Similar improvements are found when compared with the Pool at $t=150$.
- Excluding the 12 hypergraphs used for training the coarsening parameters, the EA achieves an overall reduction

in the mean final cut-size of 1.8% ($p \leq 0.05$) compared with the results achieved at $t=150$.

- Taken hypergraph-by-hypergraph, the mean final cut-size is smaller on 13 of the 18 hypergraphs. There is no significant difference compared with $t=15000$ and yet overall the average wall-clock time was $\approx 7.4\times$ faster.
- Total partitioning time with $t=150$ is of course much faster than the adaptively coarsened hypergraphs ($\approx 10\times$), however with larger cut-sizes. Thus, showing the existence of the aforementioned knee-points.

The use of a range of visual analytics tools failed to uncover any obvious relationships between the characteristics of the uncoarsened hypergraphs and the magnitude and direction of the performance difference arising from adaptive coarsening.

VII. Conclusions

Our analysis of the state-of-the-art in hypergraph partitioning algorithms reveals that despite considerable sophistication, all algorithms use a somewhat arbitrary threshold for determining the size of the initial partitioning problem to be solved. This is perhaps driven by the poor scaleability of the search algorithms involved, such as BFS.

However, experimental analysis of the 'searchability' of initial partition landscapes at different coarsening thresholds shows that larger landscapes may have properties that can be exploited by population-based search, and we derive some guidelines for algorithm design based on that analysis.

Experimental results confirm our hypothesis that there is valuable 'niche' for EA-based search that leads to statistically significant reductions in final cut-size: up to 20% compared to the default settings (Pool algorithm at $t=150$). Searching effectively in larger search spaces comes at a cost of approximately ten-fold in run-time, but this may well be warranted in many contexts such as 'one-off' design, or where subsequent processing is needed within the partitions.

Sensitivity analysis confirmed the guidelines derived from landscape analysis: recombination is useful, population size is not critical, and it is worth devoting a significant proportion of the computational budget to seeding the EA-base search.

Examining the search performance of different algorithms at different coarsenening levels, we observe that there is a 'sweet-spot' for EA-based search that is instance-dependent. We identify a novel, computationally cheap method for halting coarsening by monitoring the rate of change in information content as the hypergraph is contracted. This gives as good results as stopping at a predefined arbitrary larger threshold and with run-times reduced 7.5-fold.

We do not claim to have developed the 'best' EA to work in that niche. Rather, the aim of this paper was to establish the presence of a valuable role for EAs in hypergraph partitioning, working at a less coarsened level than currently used. In future work we will focus on (i) improved adaptive coarsening schemes, and (ii) tighter integration and re-use of information from the FM local search with the EA search processes and EDA model-building.

REFERENCES

[1] T. Lengauer, *Combinatorial algorithms for integrated circuit layout*. New York, NY, USA: John Wiley & Sons, 1990.

[2] S. Areibi and Z. Yang, "Effective memetic algorithms for VLSI design = genetic algorithms + local search + multi-level clustering," *Evol. Comput.*, vol. 12, no. 3, pp. 327–353, Fall 2004.

[3] O. Selvitopi, S. Acer, and C. Aykanat, "A recursive hypergraph bipartitioning framework for reducing bandwidth and latency costs simultaneously," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 2, pp. 345–358, Feb. 2017.

[4] A. Trifunović, "Parallel algorithms for hypergraph partitioning," Ph.D. dissertation, Department of Computing, Imperial College of Science, Technology and Medicine, University of London, London, UK, 2006.

[5] F. Lotfifar, "Hypergraph partitioning in the cloud," Ph.D. dissertation, School of Engineering and Computing Sciences, Durham University, Durham, UK, 2016.

[6] C. J. Alpert, J.-H. Huang, and A. B. Kahng, "Multilevel circuit partitioning," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 17, no. 8, pp. 655–667, Aug. 1998.

[7] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, "Multilevel hypergraph partitioning: Applications in VLSI domain," *IEEE Trans. VLSI Syst.*, vol. 8, no. 1, pp. 69–79, Mar. 1999.

[8] U. V. Çatalyürek and C. Aykanat, "Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication," *IEEE Trans. Parallel Distrib. Syst.*, vol. 11, no. 7, pp. 673–693, Jul. 1999.

[9] K. D. Devine, E. G. Boman, R. T. Heaphy, R. H. Bisseling, and U. V. Çatalyürek, "Parallel hypergraph partitioning for scientific computing," in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, P. Spirakis and H. J. Siegel, Eds. Piscataway, NJ, USA: IEEE Press, 2006, p. 10.

[10] A. Trifunović and W. J. Knottenbelt, "Parallel multilevel algorithms for hypergraph partitioning," *J. Parallel Distrib. Comput.*, vol. 68, no. 5, pp. 563–581, May 2008.

[11] U. V. Çatalyürek, M. Deveci, K. Kaya, and B. Uçar, "UMPa: A multi-objective, multi-level partitioner for communication minimization," in *Contemporary Mathematics: Graph Partitioning and Graph Clustering*, D. A. Bader, H. Meyerhenke, P. Sanders, and D. Wagner, Eds. Providence, RI, USA: AMS, 2013, vol. 588, pp. 53–66.

[12] S. Schlag *et al.*, "k-way hypergraph partitioning via n-level recursive bisection," in *Proc. ALENEX*, M. Goodrich and M. Mitzenmacher, Eds. Philadelphia, PA, USA: SIAM, 2016, pp. 53–67.

[13] T. Heuer and S. Schlag, "Improving coarsening schemes for hypergraph partitioning by exploiting community structure," in *16th Int. Symp. Experimental Algorithms, (SEA 2017)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), C. S. Iliopoulos, S. P. Pissis, S. J. Puglisi, and R. Raman, Eds., vol. 75. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017, pp. 21:1–21:19.

[14] C. M. Fiduccia and R. M. Mattheyses, "A linear time heuristic for improving network partitions," in *Proc. IEEE Design Autom. Conf.*, J. S. Crabbe, Ed. Piscataway, NJ, USA: IEEE Press, 1982, pp. 175–181.

[15] V. Osipov and P. Sanders, "n-level graph partitioning," in *Proc. Euro. Symp. Algor.*, ser. LNCS, M. de Berg and U. Meyer, Eds., vol. 6346. Berlin, Germany: Springer, 2010, pp. 278–289.

[16] Y. Akhremtsev, T. Heuer, P. Sanders, and S. Schlag, "Engineering a direct k-way hypergraph partitioning algorithm," in *Proc. ALENEX*, S. Fekete and V. Ramachandran, Eds. Philadelphia, PA, USA: SIAM, 2017, pp. 28–42.

[17] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM J. Sci. Comput.*, vol. 20, no. 1, pp. 359–392, Aug. 1998.

[18] G. Karypis, "Multilevel hypergraph partitioning," in *Multilevel Optimization in VLSICAD*, ser. Combinatorial Optimization, J. Cong and J. R. Shinnerl, Eds. New York, NY, USA: Springer US, 2003, vol. 14, ch. 3, pp. 125–154.

[19] A. J. Soper, C. Walshaw, and M. Cross, "A combined evolutionary search and multilevel optimisation approach to graph-partitioning," *J. Global Optim.*, vol. 29, no. 2, pp. 225–241, Jun. 2004.

[20] L. Kotthoff, "Algorithm selection for combinatorial search problems: A survey," *AI Mag.*, vol. 35, no. 3, pp. 48–60, Fall 2014.

[21] T. Heuer, "Engineering initial partitioning algorithms for direct k-way hypergraph partitioning," Bachelor thesis, Department of Informatics, Karlsruhe Institute of Technology, Karlsruhe, Germany, 2015.

[22] U. V. Çatalyürek and C. Aykanat, "PaToH: Partitioning tool for hypergraphs," http://bmi.osu.edu/umit/PaToH/manual.pdf, pp. 22–23, 2011.

[23] J. Kim, I. Hwang, Y.-H. Kim, and B.-R. Moon, "Genetic approaches for graph partitioning: A survey," in *Proc. GECCO*, N. Krasnogor, Ed. New York, NY, USA: ACM, 2011, pp. 473–480.

[24] U. Benlic and J. K. Hao, "A multilevel memetic approach for improving graph k-partitions," *IEEE Trans. Evol. Comput.*, vol. 15, no. 5, pp. 624–642, Oct. 2011.

[25] P. Sanders and C. Schulz, "Distributed evolutionary graph partitioning," in *Proc. ALENEX*, D. A. Bader and P. Mutzel, Eds. Philadelphia, PA, USA: SIAM, 2012, pp. 16–29.

[26] H. Meyerhenke, P. Sanders, and C. Schulz, "Parallel graph partitioning for complex networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 9, pp. 2625–2638, Sep. 2017.

[27] S. Küçükpetek, F. Polat, and H. J. Oğuztüzün, "Multilevel graph partitioning: An evolutionary approach," *J. Oper. Res. Soc.*, vol. 56, no. 5, pp. 549–562, May 2005.

[28] P. Merz and B. Freisleben, "Fitness landscapes, memetic algorithms, and greedy operators for graph bipartitioning," *Evol. Comput.*, vol. 8, no. 1, pp. 61–91, Spring 2000.

[29] A. S. Pope, D. R. Tauritz, and A. D. Kent, "Evolving multi-level graph partitioning algorithms," in *Proc. IEEE Symp. Series Comput. Intell.*, Y. Jin and S. Kollias, Eds. Piscataway, NJ, USA: IEEE Press, 2016, pp. 1–8.

[30] H. Mühlenbein and T. Mahnig, "Evolutionary optimization and the estimation of search distributions with applications to graph bipartitioning," *Int. J. Approx. Reason.*, vol. 31, no. 3, pp. 157–192, Nov. 2002.

[31] J. Schwarz and J. Očenášek, "Experimental study: Hypergraph partitioning based on the simple and advanced genetic algorithm BMDA and BOA," in *Proc. 5th Int. Mendel Conf. Soft. Comput. (MENDEL'99)*, 1999, pp. 124–130.

[32] J.-P. Kim, Y.-H. Kim, and B.-R. Moon, "A hybrid genetic approach for circuit bipartitioning," in *Proc. GECCO*, ser. LNCS, K. Deb, Ed. Berlin, Germany: Springer, 2004, vol. 3103, pp. 1054–1064.

[33] S. Coe, S. Areibi, and M. Moussa, "A hardware memetic accelerator for VLSI circuit partitioning," *Comput. Elect. Eng.*, vol. 33, no. 4, pp. 233–248, Jul. 2007.

[34] R. Andre, S. Schlag, and C. Schulz, "Memetic multilevel hypergraph partitioning," in *Proc. GECCO*, K. Takadama, Ed. New York, NY, USA: ACM, 2018, pp. 347–354.

[35] C. J. Alpert, "The ISPD98 circuit benchmark suite," in *Proc. Int. Symp. Phys. Design*, M. Sarrafzadeh, Ed. New York, NY, USA: ACM, 1998, pp. 80–85.

[36] T. A. Davis and Y. Hu, "The University of Florida sparse matrix collection," *ACM Trans. Math. Softw.*, vol. 38, no. 1, pp. 1–25, Nov. 2011.

[37] A. Belov, D. Diepold, M. Heule, and M. Järvisalo, "SAT competition 2014," http://satcompetition.org/2014/, 2014.

[38] S. S. Choi, Y. K. Kwon, and B. R. Moon, "Properties of symmetric fitness functions," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 743–757, Dec. 2007.

[39] M. Serpell and J. E. Smith, "Self-adaptation of mutation operator and probability for permutation representations in genetic algorithms," *Evol. Comput.*, vol. 18, no. 3, pp. 491–514, Fall 2010.

[40] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Jan. 2006.

[41] K. D. Boese, A. B. Kahng, and S. Muddu, "A new adaptive multi-start technique for combinatorial global optimizations," *Oper. Res. Lett.*, vol. 16, no. 2, pp. 101–113, Sep. 1994.

[42] S. Meyer-Nieberg and H.-G. Beyer, "Self-adaptation in evolutionary algorithms," in *Parameter setting in evolutionary algorithms*, ser. Studies in Computational Intelligence, F. Lobo, C. Lima, and Z. Michalewicz, Eds. Berlin, Germany: Springer, 2007, vol. 54, pp. 47–75.

[43] J. E. Smith, "Parameter perturbation mechanisms in binary coded GAs with self-adaptive mutation," in *Foundations of Genetic Algorithms 7*, C. Potta, R. Poli, J. Rowe, and K. DeJong, Eds. San Francisco, CA, USA: Morgan Kauffman, 2003, pp. 329–346.

[44] M. Z. Ali, N. H. Awad, P. N. Suganthan, and R. G. Reynolds, "An adaptive multipopulation differential evolution with dynamic population reduction," *IEEE Trans. Cybern.*, vol. 47, no. 9, pp. 2768–2779, Sep. 2017.

[45] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz, "BOA: The Bayesian optimization algorithm," in *Proc. GECCO*, D. E. Goldberg, Ed. San Francisco, CA, USA: Morgan Kaufmann, 1999, pp. 525–532.