Multilevel Refinement for Combinatorial Optimisation: Boosting Metaheuristic Performance

Chris Walshaw

Computing and Mathematical Sciences, University of Greenwich, Old Royal Naval College, Greenwich, London, SE10 9LS, UK. C.Walshaw@gre.ac.uk

1 Introduction

The multilevel paradigm as applied to combinatorial optimisation problems is a simple one, which at its most basic involves recursive coarsening to create a hierarchy of approximations to the original problem. An initial solution is found, usually at the coarsest level, and then iteratively refined at each level, coarsest to finest, typically by using some kind of heuristic optimisation algorithm (either a problem-specific local search scheme or a metaheuristic). Solution extension (or projection) operators can transfer the solution from one level to another.

As a general solution strategy, the multilevel paradigm has been in use for many years and has been applied to many problem areas (for example multigrid techniques can be viewed as a prime example of the paradigm). Overview papers such as [39] attest to its efficacy. However, with the exception of the graph partitioning problem, multilevel techniques have not been widely applied to combinatorial problems and in this chapter we discuss recent developments.

In this chapter we survey the use of multilevel combinatorial techniques and consider their ability to boost the performance of (meta)heuristic optimisation algorithms.

1.1 Fundamentals

The last 50 years have seen a huge amount of research effort devoted to the study of combinatorial optimisation problems. Spanning applied mathematics, through operations research, to management sciences, such problems are typically concerned with the allocation of resources in some way and are used in many diverse applications including scheduling, timetabling, logistics and the design of computer components.

There are now a bewildering array of (meta)heuristic optimisation algorithms (e.g. ant colony optimisation, genetic algorithms, simulated annealing, tabu search, variable neighbourhoods, etc., [4]) to address such problems and for any given application, the practitioner must often answer the question '*which is likely to be the best algorithm for my problem?*'

In considering the multilevel paradigm however, a different question occurs. Since the multilevel framework is a *collaborative* one, which always acts in concert with some other technique, the question that arises is '*given that I am using algorithm X for addressing my problem, can its performance be boosted by using a multilevel version of algorithm X?*'

As we shall see, very often the answer appears to be emphatically *yes*, either in terms of the solution quality, or the computational runtime, or both.

Indeed, in certain instances, it does seem to be the case that, in terms of performance, the decision to use a multilevel scheme is far more important then the actual choice of the underlying search algorithm.

Even more encouragingly, anecdotal evidence suggests that, in developing a full-blown multilevel scheme, the multilevel framework (which at its most basic requires a coarsening algorithm and a projection/extension operator) is generally far easier to implement than a high-quality optimisation algorithm for the problem. In fact, it is often the case that a coarsening algorithm can be built from components found in solution construction heuristics, i.e. those which are usually used to find an initial feasible (although poor quality) solution, whilst the solution extension operator is usually a trivial reversal of coarsening.

1.2 Overview

The rest of the chapter is organised as follows.

First we motivate the ideas and, via some sample results, consider evidence for the strengths of the multilevel paradigm by describing by discussing the widespread use of multilevel graph partitioning schemes. Multilevel techniques have been employed in this field since 1993 and enable very high quality solutions to be found rapidly. Thus, in Sect. 2, we provide sample results which demonstrate that the multilevel approach when used in combination with a state-of-the-art (single-level) local search strategy can dramatically improve the asymptotic convergence in solution quality.

In Sect. 3, we then survey the use of multilevel techniques, both in the field of graph-partitioning and also their increasing use in other combinatorial optimisation problems. We also look at related ideas. In Sect. 4, we look at generic features and extract some guiding principles that might aid the application of the paradigm to other problems. Finally we summarise the chapter in Sect. 5.

2 Extended Example: the Graph Partitioning Problem

The k-way graph partitioning problem (GPP) can be stated as follows: given a graph $G(V, E)$, possibly with weighted vertices and/or edges, partition the vertices into k disjoint sets such that each set contains the same (or nearly the same) vertex weight and such that the **cut-weight**, the total weight of edges cut by the partition, is minimised. In combinatorial optimisation terms, the cut-weight is the objective function whilst balancing the vertex weight is a constraint (the balance constraint) and it is well known that this problem is NP-hard.

The GPP has a number of applications, most notably the partitioning of unstructured meshes for parallel scientific computing (often referred to as mesh partitioning).

2.1 Multilevel Graph Partitioning

The GPP was the first combinatorial optimisation problem to which the multilevel paradigm was applied and there is now a considerable volume of literature about multilevel partitioning algorithms which we survey in Sect. 3.1. Initially used as an effective way of speeding up partitioning schemes, it was soon recognised as, more importantly, giving them a more 'global' perspective [21], and has been successfully developed as a strategy for overcoming the localised nature of the Kernighan-Lin (KL) [26] and other optimisation algorithms.

Typically, multilevel implementations match and coalesce pairs of adjacent vertices to define a new graph and recursively apply this procedure until the graph size falls below some threshold. The coarsest graph is then partitioned (possibly with a crude algorithm) and the partition is successively refined on all the graphs starting with the coarsest and ending with the original. At each change of levels, the final partition of the coarser graph is used to give the initial partition for the next level down.

The use of multilevel combinatorial refinement for partitioning was first proposed by both Hendrickson & Leland [19] and Bui & Jones [10], inspired by Barnard & Simon [2], who used a multilevel numerical algorithm to speed up spectral partitioning.

Fig. 1 shows an example of a multilevel partitioning scheme in action. On the top row (left to right) the graph is coarsened down to 4 vertices which are (trivially) partitioned into 4 sets (bottom right). The solution is then successively extended and refined (right to left; each graph shows the final partition for that level). Although at each level the refinement is only local in nature, a high quality partition is still achieved.

Graph Coarsening

A common method to create a coarser graph $G_{l+1}(V_{l+1}, E_{l+1})$ from $G_l(V_l, E_l)$ is the edge contraction algorithm proposed by Hendrickson & Leland [19]. The

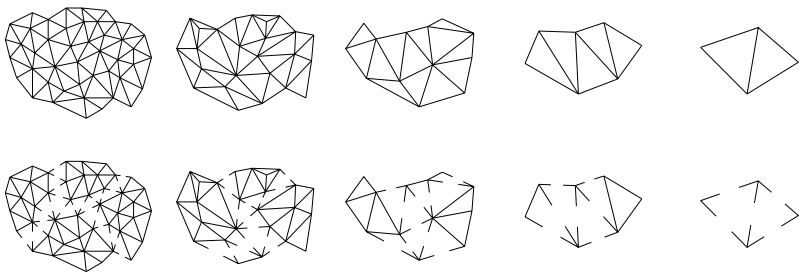

Fig. 1. An example of multilevel partitioning

idea is to find a maximal independent subset of graph edges, or a **matching** of vertices, and then collapse them. The set is independent if no two edges in the set are incident on the same vertex (so no two edges in the set are adjacent), and maximal if no more edges can be added to the set without breaking the independence criterion.

Having found such a set, each selected edge is collapsed and the vertices, $u_1, u_2 \in V_l$ say, at either end of it are merged to form a new vertex $v \in V_{l+1}$ with weight $||v|| = ||u_1|| + ||u_2||$. Edges which have not been collapsed are inherited by the child graph, G_{l+1}, and, where they become duplicated, are merged with their weight summed. This occurs if, for example, the edges (u_1, u_3) and (u_2, u_3) exist when edge (u_1, u_2) is collapsed. Because of the inheritance properties of this algorithm, it is easy to see that the total vertex weight remains the same, $||V_{l+1}|| = ||V_l||$, and the total edge weight is reduced by the sum of the collapsed edge weights.

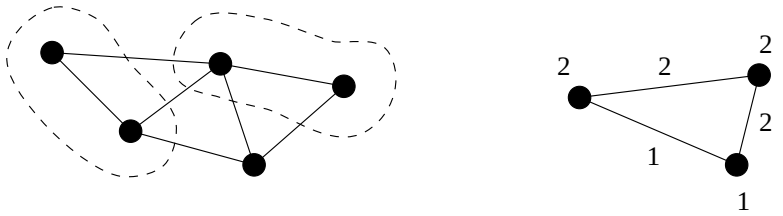

Fig. 2. An example of coarsening via matching and contraction

Fig. 2 shows an example where, on the left, two pairs of vertices are matched (indicated by a dotted line). On the right, the resulting coarsened graph is shown, with numbers illustrating the resulting vertex and edge weights (assuming that the original graph had unit weights).

A simple way to construct a maximal independent subset of edges is to create a randomly ordered list of the vertices and visit them in turn, matching

each unmatched vertex with an unmatched neighbour (or with itself if no unmatched neighbours exist). Matched vertices are removed from the list. If there are several unmatched neighbours the choice of which to match with can be random, but it has been shown by Karypis & Kumar [21] that it can be beneficial to the optimisation to collapse the most heavily weighted edges.

As discussed below (Sect. 4.1), coarsening has the effect of **filtering** the solution space. To see this suppose that two vertices $u, v \in G_l$ are matched and coalesced into a single vertex $v' \in G_{l+1}$. When a refinement algorithm is subsequently used on G_{l+1} and whenever v' is assigned to one of the partition subsets, both u and v are also both being assigned to that subset. In this way the matching restricts a refinement algorithm working on G_{l+1} to consider only those configurations in the solution space in which u and v lie in the *same* subset, although the particular subset to which they are assigned is not specified at the time of coarsening. Since many vertex pairs are generally coalesced from all parts of G_l to form G_{l+1} this set of restrictions is equivalent to filtering the solution space and hence the surface of the objective function.

The Initial Partition

The hierarchy of graphs is constructed recursively until the number of vertices is smaller than some threshold and then an initial partition is found for the coarsest graph. At its simplest, the coarsening is terminated when the number of vertices in the coarsest graph is the same as the number of subsets required, k, and then vertex i is assigned to subset S_i. However, since the vertices of the coarsest graph are not generally homogeneous in weight, this does require some mechanism for ensuring that the final partition is balanced, i.e. each subset has (approximately) the same vertex weight. Various methods have been proposed for achieving this, commonly either by terminating the contraction so that the coarsest graph G_L still retains enough vertices, $|V_L|$, to achieve a balanced initial partition (i.e. so that typically $|V_L| \gg k$) [19, 21], or by incorporating load-balancing techniques alongside the refinement algorithm, e.g. [46].

Refinement

At each level, the partition from the previous level is extended to give an initial partition and then refined. Various refinement schemes have been successfully used including a variety of metaheuristics, which we survey below, Sect. 3.1. Most commonly, however, the refinement is based on the Kernighan-Lin (KL) bisection optimisation algorithm [26] which includes the facility (albeit limited) to enable it to escape from local minima. Recent implementations almost universally use the linear time complexity improvements (e.g. bucket sorting of vertices) introduced to partitioning by Fiduccia & Mattheyses [14]. For more details see one of the many implementations, e.g. [19, 23, 46].

In terms of the multilevel framework, the only requirements are that the scheme must be able to refine an existing partition (rather than starting from

scratch) and must be able to cope with weighted graphs since, even if the original graph is not weighted, the coarsened graphs will all have weights attached to both vertices and edges because of the coarsening procedures.

Partition Extension

Having optimised the partition on a graph G_l, the partition must be extended onto its parent G_{l-1}. The extension algorithm is trivial; if a vertex $v \in V_l$ is in subset S_i then the matched pair of vertices that it represents, $v_1, v_2 \in V_{l-1}$, are also assigned to S_i.

2.2 Multilevel Results

To illustrate the potential gains that the multilevel paradigm can offer, we give some example results. These are not meant to be exhaustive in any way but merely give an indication of typical performance behaviour.

In [44], detailed tests are carried out to assess the impact of multilevel refinement on the GPP. Here we summarise those results.

The experimental data consists of two test suites, one of which is a smallish collection of 16 sparse, mostly mesh-based graphs, drawn from a number of real-life applications, and often used for benchmarking. The other test suite consists of 90 instances, originally compiled to test graph-colouring algorithms and including a number of randomly generated examples. Although perhaps not representative of partitioning applications, they reveal some interesting results. This colouring test suite is further subdivided into 3 density classes; low (under 33 %) with 58 out of 90 instances, medium (between 33 % and 67 %) with 23 instances and high (over 67 %) with just 9 instances.

In this context, the density, or **edge density**, of a graph, $G(V, E)$, is defined as the percentage of all possible edges and given by $2|E|/[|V| \cdot (|V|-1)]$, so that a **complete** graph (where every vertex is adjacent to every other), with $|V| \cdot (|V| - 1)/2$ edges, has a density of 100 %.

The tests compare the JOSTLE implementation of the Kernighan-Lin (KL) algorithm [46] against its multilevel counterpart (MLKL). As discussed in [44], and similar to most local search schemes, the KL algorithm contains a parameter, λ, known as the **intensity**, which allows the user to specify how long the search should continue before giving up (specifically how much effort the search should make to escape local minima). When $\lambda = 0$ the refinement is purely greedy in nature, whereas if $\lambda \to \infty$ the search would continue indefinitely.

To assess the algorithms, we measure the run-time and solution quality for a chosen group of problem instances and for a variety of intensities. We then normalise these values with reference solution quality and run-time values and finally plot averaged normalised solution quality against averaged normalised run-time for each intensity value.

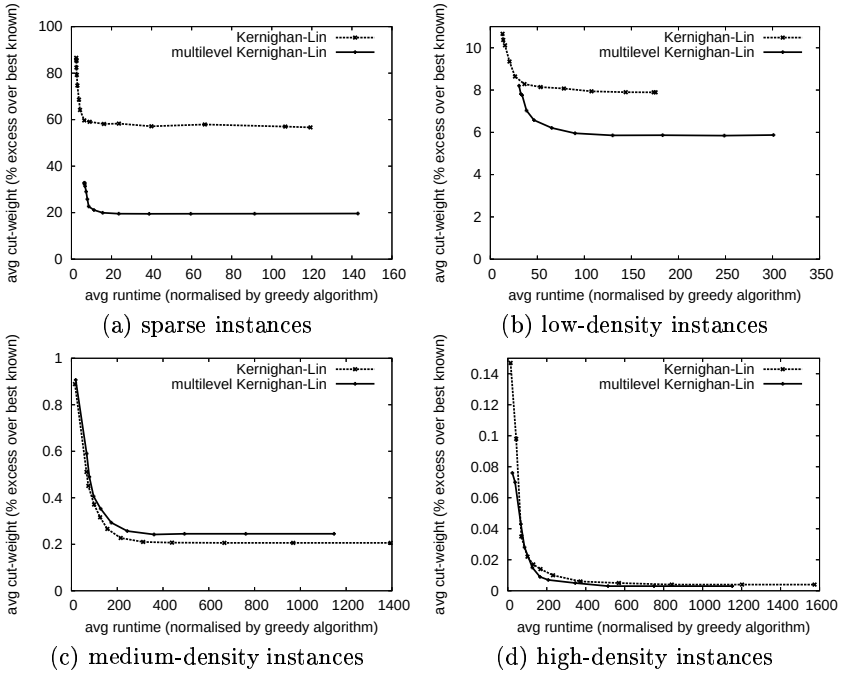

Fig. 3. Plots of convergence behaviour for the partitioning test suites

Fig. 3(a) shows the results for the sparse suite and the dramatic improvement in quality imparted by the multilevel framework is immediately clear. Even for purely greedy refinement (i.e. the extreme left-hand point on either curve) the MLKL solution quality is far better than KL and it is results like these that have helped to promote multilevel partitioning algorithms to the status they enjoy today.

Figs. 3(b)–(d) meanwhile show the partitioning results for the colouring test suite. Fig. 3(b) more or less confirms the conclusions for the sparse results and although the curves are closer together, MLKL is the clear winner. For the medium and high-density examples however, it is a surprise (especially considering the widely accepted success of multilevel partitioning) to find that these conclusions are no longer valid. For the high-density instances, Fig. 3(d), MLKL is still the leading algorithm, although only very marginally. However for the medium-density results, Fig. 3(c), MLKL fails to achieve the same performance as KL and the multilevel framework appears to actually hinder the optimisation. We discuss this further in the following section.

2.3 Iterated Multilevel Results

Although the medium density results are disappointing, in fact a simple resolution does exist which works by reusing the best partitions that have been found. Indeed, given any partition of the original problem we can carry out solution-based **recoarsening** by insisting that, at each level, every vertex v matches with a neighbouring vertex in the same set. When no further coarsening is possible this will result in a partition of the coarsest graph with the same cost as the initial partition of the original. Provided the refinement algorithms guarantee not to find a worse partition than the initial one, the multilevel refinement can then guarantee to find a new partition that is no worse than the initial one.

This sort of technique is frequently used in graph-partitioning for dynamic load-balancing, e.g. [36, 45], although if the initial partition is unbalanced, the quality guarantee can be lost in satisfying the balance constraint. However it can also be used to find very high quality partitions, albeit at some expense, and the multilevel procedure can be iterated via repeated coarsening and uncoarsening. At each iteration the current best solution is used to construct a new hierarchy of graphs, via recoarsening, and guarantees not to find a worse solution than the initial one. However, if the matching includes a random factor, each iteration is very likely to give a different hierarchy of graphs to previous iterations and hence allows the refinement algorithm to visit different solutions in the search space.

We refer to this process as an **iterated multilevel algorithm** (see Sect. 3.1 for further discussion). It requires the user to specify an additional intensity parameter, namely the number of failed outer iterations (i.e. the number of times the algorithm coarsens and uncoarsens the graph without finding a better solution).

Fig. 4 illustrates the results for the iterated multilevel algorithm (IMLKL) alongside the MLKL and KL results for the two test suites. These plots contain exactly the same information about MLKL and KL as Fig. 3, only here it is more compressed because of the long IMLKL run-times.

In fact the results for the sparse and high-density instances are not so interesting; for the sparse suite, Fig. 4(a), IMLKL more or less continues the MLKL curve in Fig. 3(a) with a few percentage points improvement and very shallow decay, whilst for the high-density instances, Fig. 4(d), IMLKL does not appear to offer much improvement at all. However for the low and medium-density subclasses, in Figs. 4(b) and 4(c) respectively, the asymptotic performance offered by IMLKL is impressive and worthy of further and more thorough investigation. In both cases IMLKL dramatically improves on MLKL and, for the medium-density instances, even appears to overcome the shortcomings of MLKL and significantly improves on the KL results.

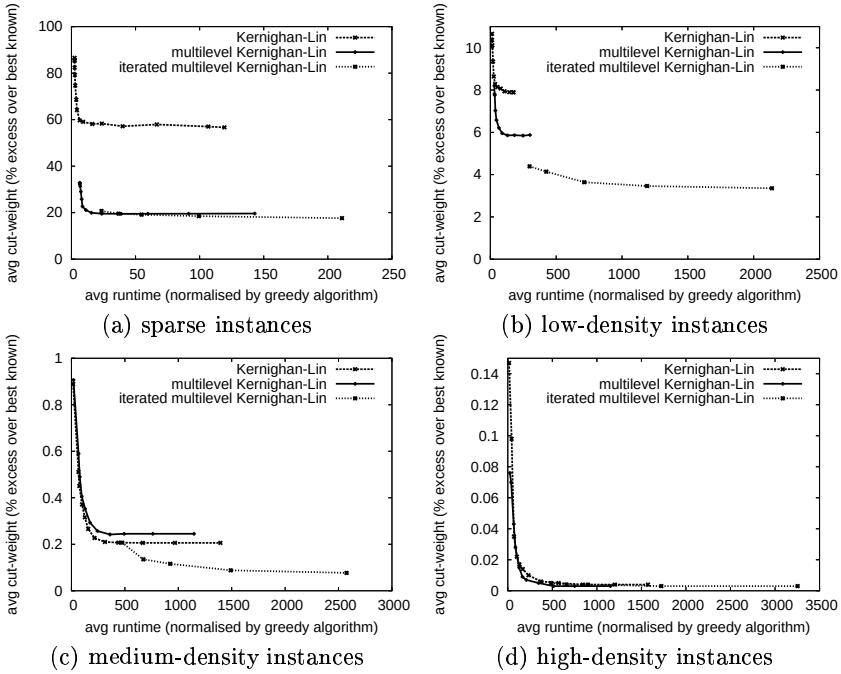

Fig. 4. Plots of convergence behaviour including iterated multilevel partitioning results

Summary

It is clear that the multilevel framework can enormously benefit the performance of a state-of-the-art local search scheme. Its task is made more difficult if the graphs are dense, but nevertheless, via the simple technique of recoarsening, an iterated multilevel scheme seems to be able to overcome this.

3 Survey of Multilevel Implementations

In this section we survey existing literature about multilevel implementations in a variety of contexts.

The survey is broken up into three categories: graph-partitioning, other combinatorial problems and related ideas. As mentioned above, multilevel graph-partitioning has been widely investigated and so Sect. 3.1 affords a number of general observations about multilevel strategies. Indeed it is because of its success in this area that the multilevel paradigm has started to be utilised elsewhere and so Sect. 3.2 takes a look at the increasing numbers of multilevel algorithms for other combinatorial problems. Finally Sect. 3.3 considers similar concepts such as smoothing.

3.1 Survey of Multilevel Graph-Partitioning

As has been mentioned in Sect. 2, graph-partitioning has been by far the most common application area for multilevel refinement and it is now a *de facto* standard technique to use. Furthermore, it has been tested with a wide variety of metaheuristic and local refinement schemes and gives a good indication of how robust the multilevel framework can be. In addition, a number of enhancements and extensions have appeared, all of which demonstrate the flexibility of the paradigm.

Table 1. Multilevel graph-partitioning algorithms and variants

Feature of interest	**References**
Multilevel metaheuristics	
Ant Colony Optimisation	Langham & Grant [28]; Korošec *et al.* [27]
Cooperative Search	Toulouse *et al.* [40]
Genetic Algorithms	Kaveh & Rahimi [25]; Soper *et al.* [38]
Tabu Search	Battiti *et al.* [3]; Vanderstraeten *et al.* [41]
Simulated Annealing	Romem *et al.* [32]; Vanderstraeten *et al.* [41]
Multilevel enhancements	
Coarsening Homogeniety	Abou-Rjeili & Karypis [1]; Gupta [17]
Constraint Relaxation	Walshaw & Cross [46]
Inexact Coarsening	Bui & Jones [10]
Recoarsening	Gupta [17]; Toulouse *et al.* [40]; Walshaw [44]
Related problems	
Aspect Ratio Optimisation	Vanderstraeten *et al.* [41]; Walshaw *et al.* [48]
Hypergraph Partitioning	Karypis & Kumar [24]
Multi-constraint/objective	Karypis & Kumar [22]; Schloegel *et al.* [37]
Network Mapping	Walshaw & Cross [47]

Table 1 shows a summary of some of the implementations. This list is far from comprehensive; indeed a survey of multilevel graph-partitioning algorithms could fill a paper in itself. However, here the aim is to highlight a variety of interesting cases and we focus on three different classes: **multilevel metaheuristics**, **multilevel enhancements** and **related problems** that have been successfully addressed by multilevel algorithms.

Multilevel Metaheuristic Refinement Schemes

A first point to note is the variety of optimisation algorithms with which the multilevel framework has been used. Apart from the commonly used problem-specific algorithm of Kernighan & Lin (KL) [26], mentioned above, a wide variety of well-known metaheuristics have been applied. These are normally

used to refine the solution at each level and, provided they can operate on weighted graphs, essentially require no modifications. Specifically, ant colony optimisation [28, 27], cooperative search [40], evolutionary/genetic algorithms [25], simulated annealing [32, 41] and various flavours of tabu search [3, 41], have been successfully applied, generally with great success.

Although space precludes a more detailed study, positive comments by authors are common. For example, Vanderstraeten *et al.*, referring to multilevel simulated annealing and multilevel tabu search, state that the '*contraction procedure not only speeds up the ... partitioning method, but also results in better mesh decompositions*' [41]. Meanwhile, Korošec *et al.*, in their implementation of multilevel ant colony optimisation, state that '*with larger graphs, which are often encountered in mesh partitioning, we had to use a multilevel method to produce results that were competitive with the results given by other algorithms*' [27].

Multilevel Enhancements

Perhaps of more interest, especially in terms of applying the multilevel framework to completely different problem areas, is the development of a number of multilevel enhancements which can be used to improve performance still further.

For example, a number of authors have noted that the coarsest graphs can become very inhomogeneous in terms of vertex weight [1, 17]. This is particularly noticeable for so-called **scale-free graphs**[1], which often arise when modelling links between web pages. In the worst cases, the coarsening may start to form star graphs, with one very large 'super-vertex' attached to many small vertices. Most coarsening schemes cannot cope with this type of graph and will only manage to collapse one edge at each level (i.e. a multilevel scheme for a star graph with 1,000 vertices will have 1,000 levels) resulting in very poor runtime performance. As a result, coarsening algorithms have been introduced to ensure that the vertex weights stay relatively homogeneous, usually by merging clusters of two (or more) vertices which are not adjacent, e.g. [1, 17]. We refer to this as **coarsening homogeneity** and although it might seem very specific to partitioning, a similar idea has recently been used in a multilevel algorithm for the Vehicle Routing Problem [30] (see below, Sect. 3.2).

Another multilevel enhancement is developed in [46], where it is demonstrated that by relaxing the balancing constraint (i.e. that the subsets are all of the same size) at the coarsest graph levels, and tightening it up level by level, higher quality results can be found. This idea of **constraint relaxation** has also been used for the Vehicle Routing Problem [30] (see also Sect. 3.2).

[1] Typically in scale-free graphs the degree distribution follows a power-law relationship so that some vertices are highly connected, although most are of low degree.

In one of the first multilevel graph partitioning implementations [10], Bui & Jones collapsed edges and merged vertices without taking account of vertex or edge weights (see Sect. 2.1). As a result, the coarsened graph did not represent the original problem with complete accuracy (e.g. since an edge in a coarsened graph might equally represent one or one hundred edges of the original graph and so it is no longer obvious which cut edges to prefer). We refer to this as **inexact** coarsening and although it is not necessarily recommended, especially in the case of graph partitioning where it is fairly easy to create **exact** coarse representations of the problem, it is interesting to see that this kind of coarsening could still '*dramatically improve the performance of the Kernighan-Lin and greedy algorithms*' [10]. This is especially encouraging for other combinatorial problems, where it may not even be possible to create an exact coarsened representation.

Finally, as we have seen in Sect. 2.3, a powerful, and very straightforward technique is to use solution-based **recoarsening**. Indeed this feature can be used in two different ways: either, as in [44], as an outer loop which treats the multilevel partitioner almost as a black-box and iterates its use; or, as in [17], where the framework cycles up and down through the coarser levels to find a very high quality initial solution for very little computational expense (since the coarse graphs are very small). In Sect. 3.2, we classify these as either **external** or **internal** recoarsening, respectively.

Related Partitioning Problems

The multilevel paradigm has also been successfully extended to a range of partitioning problem variants.

In particular, the development of successful multilevel hypergraph[2] partitioning schemes has sparked a great deal of interest in the VLSI/circuit partitioning community, e.g. [11]. Indeed, Karypis and Kumar, pre-eminent developers of such schemes state that '*the power of [their algorithm] is primarily derived from the robustness of the multilevel paradigm that allows the use of a simple k-way partitioning refinement heuristic instead of the [commonly used] $O(k^2)$ complexity k-way FM refinement*' [24].

Similarly, in [22, 37], multilevel graph partitioning techniques have been successfully modified to deal with other, more generalised graphs such as those which have multiple vertex weights (multi-constraint problems) or multiple edge weights (multi-objective problems).

Finally and more specifically, multilevel graph partitioning techniques have been shown to adapt well to modified objective functions. For example, in [41, 48], the aspect ratio (shape) of the subdomains is optimised as an alternative to the cut-weight. Meanwhile, [47] considers the problem mapping graphs onto parallel networks with heterogeneous communications links – a successful

[2] A hypergraph is a generalisation of a graph in which an edge describes a relationship between an arbitrary number of vertices.

mapping is then one in which adjacent subsets generally lie on 'adjacent' processors.

An important point about these last two problems is that, in both cases, the coarsening scheme requires almost no modifications and the new problem class is optimised solely by changes to the objective function (and hence the refinement scheme). This makes an interesting point: using the multilevel framework, the *global* layout of the final partition can be radically changed just by modifying the *local* cost function. This corroborates the suggestion that the multilevel framework adds a global perspective to (local) partitioning schemes.

3.2 Survey of Multilevel Combinatorial Optimisation

Apart from its wide uptake in the graph-partitioning area, multilevel refinement schemes are increasingly appearing for other combinatorial problems. Typically, the results are early indicators and the ideas not yet fully developed, but in most applications authors report successes, either in speeding up a metaheuristic or local search algorithm, or in improving the quality of its results. Furthermore, it is often seen as an appropriate method for addressing larger-scale problem instances which, hitherto, have not been tractable.

To survey these implementations, we first describe each one briefly, highlighting the multilevel techniques that it employs, and then, at the end of this section, attempt to summarise and classify the approaches.

Biomedical Feature Selection. In [29], Oduntan investigates multilevel schemes for feature selection and classification in biomedical data. Two possible coarsening strategies are suggested, one of which clusters decision variables, in a manner akin to multilevel partitioning, and represents them as a single variable and another, 'feature pre-setting', which recursively excludes features (decision variables) from each level. This second approach is unlike most multilevel implementations in that it is essentially evaluating only **partial** solution spaces at each level (as opposed to **restricted** spaces), but, because of the computational cost of maintaining an exact cost evaluation for the clustering version, random feature pre-setting is used for most of the experimentation. Nonetheless, the multilevel technique is '*outstandingly better*' and '*generates higher and more stable average classification accuracies*' than the other evaluated feature selection techniques.

Capacitated Multicommodity Network Design (CMND). In [12], Crainic *et al.* implement a multilevel cooperative search algorithm for the CMND problem. The coarsening is achieved via fixed edges which are computed from an initial solution calculated by a tabu search (solution-based coarsening) and a form of iterated multilevel refinement is performed by propagating elite solutions up through the multilevel hierarchy (as well as down). The authors discuss the issue of how rapidly to coarsen the problem and the compromise between a high number of levels and large coarsening factors, and cite this as a future research challenge. Nonetheless, the initial results are very

encouraging and experiments on a set of benchmark problems showed that the approach yields '*solutions comparable to those obtained by the current best metaheuristics for the problem*' and that the multilevel framework '*appears to perform better when the number of commodities is increased (which, normally, increases the difficulty of the problem)*'.

Covering Design. Another problem that has recently been investigated with multilevel cooperative techniques is that of covering design in which the aim is to minise the number subsets of a given size which 'cover' the problem instance (i.e. so that every member of the problem instance is contained in at least a given number of the subsets). This problem is extremely difficult to solve to optimality and in [13], Dai *et al.* use multilevel cooperative tabu search to improve upper bounds (in order, for example, to improve the efficiency of enumeration techniques such as branch-and-cut). They use solution-based re-coarsening and a 'direct interpolation' operator, which projects elite solutions found at the coarsest level directly down to the the original problem. The results are impressive – the methods succeed in computing new upper bounds for 34 out of 158 well-known benchmark problems – and the authors confirm that, for this problem, '*multilevel search and cooperation drastically improve the performance of tabu search*'. Furthermore, since similar cost functions and neighbourhood structures exist in other problems such as packing design, t-design and feature selection in bio-informatics and data mining, the authors suggest that '*the main framework of the re-coarsening and direct interpolation operations can be applied in a multilevel cooperative search algorithm for such problems*'.

DNA Sequencing by Hybridization. Blum & Yábar use a multilevel framework to enhance an ant colony algorithm for the computational part of this problem [5, 6]. It is closely related to the selective travelling salesman problem, a variant in which only a subset of the vertices are visited, and the authors use a similar coarsening approach to that mentioned below, in which edges are fixed to create tour segments or paths. In experiments, the algorithms (both single- and multilevel) solve all benchmark problems to optimality, but the multilevel version substantially reduces the computation time of the ant colony optimisation, and is between 3 to 28 times faster. Furthermore the authors believe that for larger problem instances, the multilevel framework '*may also improve the quality of the obtained solutions*'.

Graph Colouring Problem. In [44], Walshaw discusses the development of multilevel graph colouring schemes. The coarsening uses similar vertex matching to the graph partitioning, with one important difference: since the objective here is to colour each vertex differently to any of its neighbours, the matching is carried out between *non-adjacent* vertices (so that all vertices in a coarsened cluster can be given the same colour). The experimentation illustrates that, for sparse and low-density graphs at least, the multilevel paradigm can either speed up or even give some improvements in the asymptotic convergence of well known algorithms such as tabu search. However, the impact of the multilevel framework is less impressive than for other problems and the

paper concludes with a number of suggestions that might help to improve the results.

Graph Ordering. One problem area that has received a fair amount of interest from multilevel practitioners is that of graph ordering. This is almost certainly because the applications from which it arises (such as the efficient solution of linear systems) are closely related to those in which graph partitioning occurs (e.g. parallel scientific computing) and the two problems have some overlap.

For example, Boman & Hendrickson describe the use of a multilevel algorithm for reducing the envelope of sparse matrices [7], a technique which aims to place all the non-zeroes as close as possible to the diagonal of a matrix and which can help to speed up the solution of sparse linear systems. They report good results, better than some of the commonly used methods, although they conclude that their scheme would probably be better if combined with a state-of-the-art local search algorithm. This conclusion is confirmed by Hu & Scott who have also developed a multilevel method for the same problem and which uses such a scheme, the hybrid Sloan algorithm, on the coarsest graph only [20]. They report results which are of similar quality to the standalone hybrid Sloan algorithm (i.e. as good as the best known results) but which, on average, can be computed in half the time.

Both of these approaches use a coarsening algorithm based on vertex matching, and similar to those used for multilevel graph partitioning schemes. More recently, however, Safro *et al.* discuss the use of a multilevel algorithm for the linear arrangement problem [35], which has the aim of ordering the vertices of a graph so that the sum of edge lengths in the corresponding linear arrangement is minimised, a problem with several diverse applications. In this work, the coarsening is described as 'weighted aggregation' (as opposed to 'strict aggregation') in which each vertex can be divided into fractions which are then merged. In other words, a coarsened vertex will in general consist of fractions of several vertices from the original graph, rather than a discrete set of two (or more). In experiments, the authors found that their algorithm generally outperformed standard methods on most test cases, although it had some problems on uniform random graphs. This echoes some of the findings for multilevel graph partitioning which seems to prefer certain types of graph.

Finally, it is worth noting that unlike most problems discussed in this section, the schemes described in all three papers produce inexact representations of the problem in coarsened spaces. This is due to the nature of the graph ordering problem and may be inevitable, but see the classification section below (page 17) for further discussion.

Travelling Salesman Problem (TSP). The TSP is a prototypical (arguably *the* prototypical) combinatorial problem and consequently is one of the first on which new algorithms are tested.

In the earliest multilevel approach to the TSP, Saab, using a technique described as 'dynamic contraction', applied algorithms to recursively construct chains of cities and then implemented a simple local search algorithm [34].

The results demonstrate that '*the improvement due to contraction*' (in our context, the improvement due to the multilevel framework) '*is significant in all cases and ranges from 33.1 % to 108.4 %*' although possibly, as the author acknowledges, this is because the local search is a very basic heuristic in this case. This paper is also interesting because it proposes dynamic contraction as a generic strategy for combinatorial problems and, apart from the TSP, also demonstrates the approach on the graph bisection problem.

Subsequently, Walshaw independently applied a similar coarsening approach using fixed edges to build a multilevel version of the well-known chained Lin-Kernighan algorithm (together with a more powerful variant, Lin-Kernighan-Helsgaun) [42, 44]. The multilevel results showed significant improvement on the single-level versions and are '*shown to enhance considerably the quality of tours for both the Lin-Kernighan and Chained Lin-Kernighan algorithms, in combination the TSP champion heuristics for nearly 30 years*'.

Finally, in [8], Bouhmala combined a multilevel approach with a genetic algorithm adapted for the TSP. Interestingly, and in contrast to the two previous approaches, the coarsening is based on merging pairs of cities and averaging their coordinates. Although this is a simpler approach to fixing edges, it means that any coarsened version does not represent the original problem instance exactly and, because of the modified coordinates, the cost of a solution in one of the coarsened spaces is not the same as the cost if that solution is projected back to the original. Nonetheless, the results indicate that '*the new multilevel construction algorithm*' (i.e. just using the coarsening without any refinement) '*produces better results than the Clark-Wright algorithm, and that the multilevel genetic algorithm was found the clear winner when compared to the traditional genetic algorithm*'.

Vehicle Routing Problem (VRP). A problem closely related to the TSP (although significantly complicated by additional side constraints) is the VRP and in [30], Rodney *et al.* demonstrate that it too is susceptible to multilevel techniques. The coarsening scheme is similar to that used previously for the TSP [34, 42] and recursively fixes edges into potential routes (whilst additionally respecting vehicle capacity constraints).

The resulting framework is tested with a variety of problem-specific local search heuristics and it is found that, for a test suite of well-known benchmark problems, the best multilevel solutions are on average within 7.1 % of optimality as compared with 18.4 % for the single-level version of the algorithm. Although this is not quite so good as results from a state-of-the-art genetic algorithm code, it has since been shown that an iterated version (see below and Sect. 2.3) of the multilevel scheme finds solutions within 2.6 % of optimal, comparable with the genetic algorithm and considerably faster (especially as the problem size increases).

As part of this work on the VRP, two multilevel enhancement techniques are developed and tested.

The first employs **constraint relaxation** (referred to in [30] as 'route overloading'), as discussed in Sect. 3.1. In the context of the VRP, this means

allowing routes in the coarser level to exceed the vehicle capacity constraint. This is then tightened up gradually as the multilevel scheme approaches the finer levels and, ultimately, the original problem.

The second enchancement (referred to as 'segment balancing') tried to balance the fixed edge segments in terms of demand and cost in order to provide more **homogeneous coarsening** (see also Sect. 3.1).

Combined together, these two enhancements provide the best results and are useful ideas in the generic application of multilevel techniques.

Other Problems. The above list is not comprehensive and, for example, multilevel techniques have been applied to other problems such as satisfiability [33] and the still life problem [15]. Unfortunately, neither example showed significant benefits, either because the work is only in its preliminary stages and still requires further development, or possibly because the problem is simply not suitable for multilevel coarsening. In either case, it seems appropriate to exclude it from generalisations about the multilevel framework pending further investigation (since, as is common with heuristics, we merely seek evidence that multilevel techniques can be used successfully, rather than proof that they will always be successful). However, it is worth noting, as is discussed in [44], that although the multilevel paradigm seems to assist in most problems to which it has been applied, the benefits are not necessarily universal.

Summary and Classification

Because of the flexibility of the paradigm, multilevel schemes can appear in a variety of guises. Nonetheless, it is still possible to draw out some common features from the schemes listed above. Table 2 summarises the above approaches and aims to help classify them, both by the optimisation algorithm they use for refinement, and in terms of the multilevel techniques they employ: the coarsening strategy (**coarsen**); whether they produce **exact** coarse representations of the problem; and what type, if any, of recoarsening they use (**recoarsen**).

One of the most obvious classifications then is the **refinement** algorithm used to optimise the solution at each level. Often these are special-purpose or **problem-specific**, but in many cases a 'standard' metaheuristic is used (although it should be understood that in the context of combinatorial optimisation there is no such thing as a standard metaheuristic and all will require some modification or, at the very least, parameter tuning).

A second way of classifying the schemes is by their coarsening strategy (the column headed **coarsen**). Although this is always problem-specific, it will generally depend on whether the problem requires a solution based on an ordering or a classification of the vertices. In the case of ordering problems (such as the travelling salesman and vehicle routing problems), it is usual to recursively fix edges between vertices, a **path**-based coarsening. For classifi-

Table 2. Multilevel algorithms for combintorial problems

Problem, [reference]	**refinement**	**coarsen**	**exact**	**recoarsen**
Biomedical feature selection [29]	Tabu Search	set$^+$	yes	
Capacitated multicommodity network design [12]	Cooperative Search	path	yes	internal
Covering design [13]	Cooperative Search	set	yes	internal
DNA sequencing [5, 6]	Ant Colonies	path	yes	
Graph colouring [44]	Tabu Search	set	yes	
Graph ordering (envelope) [7]	problem-specific	set	no	
Graph ordering (wavefront) [20]	problem-specific	set	no	
Graph ordering (linear) [35]	Simulated Annealing	set*	no	external
Travelling salesman [34]	problem-specific	path	yes	
Travelling salesman [42]	problem-specific	path	yes	
Travelling salesman [8]	Genetic Algorithms	set	no	
Vehicle routing [30]	problem-specific	path	yes	external

cation problems (such as graph partitioning and graph colouring), it is usual to merge vertices, a **set**-based approach.

Although it might seem that these are very similar operations, in practice path-based coarsening generates a set of fixed path-segments in the coarsened spaces and each path-segment contains an ordering of its internal vertices. Meanwhile, set-based coarsening usually generates a weighted graph; edges that are internal to each merged set of vertices are collapsed.

Note that two variant approaches were also employed: in biomedical feature selection [29], as an alternative to set-based clustering the implementation also coarsens by recursively excluding decision variables and hence evaluating only partial solutions, denoted as 'set$^+$' in the table; meanwhile the multilevel approach to the linear graph ordering problem [35] uses an aggregated set-based approach, denoted as 'set*' in the table – see above for details.

A third way of comparing schemes is by considering whether coarsened versions of the problem instance give an **exact** or an **inexact** approximation of the original solution space. In this context, by exact we mean that evaluation of the objective function for a solution of a coarsened space is exactly the same as if that solution were extended back into the original space and the objective function evaluated there.

Note that even for problems where exact representation is possible, it is usually also possible to generate inexact representations. As we have seen in Sect. 3.1, at its simplest this can just be by ignoring vertex and or edge weights when coarsening graphs for partitioning purpose. Alternatively, if nearby vertices are merged to create a coarsened version of a travelling salesman problem (as mentioned above), then the inter-vertex distances will be incorrect and hence the coarsed problem is inexact.

Usually, employing an exact coarsening is more accurate but will involve some modifications to the optimisation scheme (for example, to take account

of vertex weights or fixed edges). Conversely, an inexact coarsening is easier to implement and the optimisation scheme can often be used without modification.

Interestingly, all of the approaches to graph ordering use set-based coarsening and merge vertices, rather than employing the path-based fixed edges that might be expected from an ordering problem, and as a result they all end up with an inexact representation of the problem. It is intriguing to ask whether an exact representation might be more effective in these cases.

A final method we have used to classify the work is to look at whether the approach employs **recoarsening** (the column labelled **recoarsen**), possibly as part of an iterated multilevel scheme. As we have seen in Sect. 2.3 this can be extremely easy to implement and highly effective. However we also distinguish here between **external** and **internal** recoarsening.

In this context, internal recoarsening can propagate elite solutions *up* through the multilevel hierarchy (as well as down) and the scheme may only ever carry out one refinement phase on the original uncoarsened problem (i.e. once it believes that a very high quality solution has been found at the coarser levels).

Conversely, external recoarsening takes place as part of repeated multilevel cycles (see Fig. 6), in an iterated multilevel scheme (in this case refinement will take place on the original uncoarsened problem at the end of every cycle). Of course, the distinction is not completely clear and it is possible for a scheme using internal recoarsening to return repeatedly to the original problem and thus resemble an iterated scheme.

3.3 Other Related Work

Because multilevel algorithms are well-known in many areas of mathematics other than combinatorial optimisation, there is a large body of literature which could be said to be related to the methods presented here. In particular, **multigrid** methods are often used to solve partial differential equations on a hierarchy of grids or meshes, whilst **multi-scale** or **multi-resolution** methods typically address continuous problems by viewing them at a number of different levels. However, because of the nature of the problems the operators which transfer solutions from one scale to another are necessarily somewhat different from the discrete techniques discussed here. For interested readers a good start is Brandt's review paper [9] and for an analysis of the fundamental similarities of all these ideas see Teng [39].

Another related idea is that of **aggregation** which can be used either to approximate an intractable problem with a smaller one or, sometimes, to provide decision-makers with models at different levels of detail. In that sense, it tends to deal more with the modelling aspects of optimisation problems (as opposed to finding a solution for a given model). However, it is certainly true that, when combined with **disaggregation** there is a degree of crossover with

multilevel ideas. For further information see the survey paper of Rogers *et al.* [31].

An idea particularly related in scope and design to the principles behind multilevel refinement is the search space smoothing scheme of Gu and Huang [16]. This uses recursive smoothing (analogous to recursive coarsening) to produce versions of the original problem which are simpler to solve. Thus in the example application Gu and Huang apply their technique to the TSP by forcing the inter-city edges to become increasingly uniform in length at each smoothing phase (if all edges between all cities are the same length then every tour is optimal). The obvious drawback is that each smoothing phase distorts the problem further (so that a good solution to a smoothed problem may not be a good solution to the original). In addition, the smoothed spaces are the same size as the original problem, even if the solution is potentially easier to refine, and hence may be equally as expensive to optimise. By contrast, exact multilevel coarsening filters the solution space (although with the obvious drawback that the best solutions may be removed from the coarsened spaces) and so the coarsened spaces are smaller and hence can be refined more rapidly (even inexact coarsening reduces the size of the space although strictly it does not filter it). It is also unclear whether search space smoothing is as general as coarsening and hence whether it could be applied to problems other than the TSP.

Finally, multilevel combinatorial optimisation is also closely related to developments in various semi-discrete optimisation problems. For example, it has been applied, with great success, to force-directed (FD) graph drawing. This is not a combinatorial problem, since the optimisation typically minimises a continuous energy function, but it does share some of the characteristics. Until recently FD methods were generally limited to small, sparse graphs, typically with no more than 1,000 vertices. The introduction of the multilevel framework, however, extended the size of graphs to which they could successfully be applied by several orders of magnitude, again through the 'global' improvement given by the multilevel scheme [18, 43].

4 Generic Analysis of the Multilevel Framework

In this section we draw together common elements of the examples in the previous sections. We give an explanation for the strengths of the multilevel paradigm and derive some generic guidelines for future attempts at other combinatorial problems.

4.1 Multilevel Dynamics

As we have seen, the multilevel paradigm is a simple one, which at its most basic involves recursive coarsening to create a hierarchy of approximations to

the original problem. An initial solution is found and then iteratively refined, usually with a local search algorithm, at each level in reverse order.

Considered from the point of view of the hierarchy, a series of increasingly coarser versions of the original problem are being constructed. It is hoped that each problem P_l retains the important features of its parent P_{l-1} but the (usually) randomised and irregular nature of the coarsening precludes any rigorous analysis of this process.

On the other hand, viewing the multilevel process from the point of view of the objective function and, in particular the hierarchy of solution spaces, is considerably more enlightening. Typically the coarsening is carried out by matching groups (usually pairs) of solution variables together and representing each group with a single variable in the coarsened space.

Previously authors have made a case for multilevel schemes (and in particular partitioning) on the basis that the coarsening successively *approximates* the problem with smaller, and hence easier to solve, solution spaces.

In fact it is somewhat better than this; as we have seen in the discussion about coarsening for the graph-partitioning problem (GPP), Sect. 2.1, provided that the coarsening is exact, it actually *filters* the solution space by placing restrictions on which solutions the refinement algorithm can visit.

A similar argument can be made for path-based coarsening algorithms, such as those employed for the TSP [44], and in a more general sense, we can think about combining decision variables so that changing one decision variable in a coarsened space is equivalent to changing several in the original solution space. In all cases, provided that the coarsening is exact, then coarsening has the effect of filtering the solution space.

Furthermore, an investigation of how the filtering actually performs for the GPP & TSP is carried out in [49] and it is shown that typically the coarsening filters out the higher cost solutions at a much faster rate than the low cost ones, especially for sparse and low-density problems.

We can then hypothesise that, if the coarsening manages to filter the solution space so as to gradually remove the irrelevant high cost solutions, the multilevel representation of the problem combined with even a fairly simple refinement algorithm should work well as an optimisation strategy. And when combined with sophisticated metaheuristics, it can be very powerful indeed.

On a more pragmatic level this same process also allows the refinement to take larger steps around the solution space (e.g. for graph partitioning, rather than swapping single vertices, the local search algorithm can swap whole sets of vertices as represented by a single coarsened vertex). This may be why the strategy still works even if the coarsening is inexact.

One further general observation about multilevel dynamics, that can be drawn from experimental evidence, is that the multilevel framework often seems to have the effect of stabilising the performance of the local search schemes. In particular, for the graph partitioning and travelling salesman problems (and the GCP at low intensities) the multilevel versions appear to have much lower variation in solution quality (in terms of the standard

deviation of randomised results) [44]. However, it is not clear why this should be the case.

4.2 A Generic Multilevel Framework

To summarise the paradigm, multilevel optimisation combines a coarsening strategy together with a refinement algorithm (employed at each level in reverse order) to create an optimisation metaheuristic. Fig. 5 contains a schematic of this process in pseudo-code (here P_l refers to the coarsened problem after l coarsening steps, C_l is a solution of this problem and C_l^0 denotes the initial solution).

```
multilevel refinement( input problem instance P_0 )
begin
    l ← 0          // level counter
    while (coarsening)
        P_{l+1} ← coarsen( P_l )
        l ← l+1
    end
    C_l = initialise( P_l )
    while (l > 0)
        l ← l−1
        C_l^0 ← extend( C_{l+1}, P_l )
        C_l ← refine( C_l^0, P_l )
    end
    return C_0   // solution
end
```

Fig. 5. A schematic of the multilevel refinement algorithm

The question then arises, how easy is it to implement a multilevel strategy for a given combinatorial problem?

First of all let us assume that we know of a refinement algorithm for the problem, which refines in the sense that it attempts to improve on existing solutions. If no such refinement algorithm exists (e.g. if the only known heuristics for the problem are based on construction) it is not clear that the multilevel paradigm can be applied.

Typically the refinement algorithm will be either a problem-specific algorithm, or a metaheuristic, and it must be able to cope with any additional restrictions placed on it by the coarsening (e.g. for the set-based coarsening, the graphs are always weighted whether or not the original is; for path-based, the refinement must not change fixed edges in the coarsened levels).

To implement a multilevel algorithm, given a problem and a refinement strategy for it, we then require three additional basic components: a coarsening

algorithm, an initialisation algorithm and an extension algorithm (which takes the solution on one problem and extends it to the parent problem). It is difficult to talk in general terms about these requirements, but the existing examples suggest that the extension algorithm can be a simple and obvious reversal of the coarsening step which preserves the same cost.

The initialisation is also generally a simple canonical mapping where by canonical we mean that a solution is 'obvious' (e.g. GPP – assign k vertices one each to k subsets; GCP – colour a complete graph; TSP – construct a tour to visit 2 cities) and that the refinement algorithm cannot possibly improve on the initial solution at the coarsest level (because there are no degrees of freedom).

Coarsening

This just leaves the coarsening which is then perhaps the key component of a multilevel implementation. For the existing examples three principles seem to hold:

- The number of levels need not be determined *a priori* but coarsening should cease when any further coarsening would render the initialisation degenerate.
- A solution in any of the coarsened spaces should induce a legitimate solution on the original space. Thus, at any stage after initialisation the current solution could simply be extended through all the problem levels to achieve a solution of the original problem.
- Ideally, any solution in a coarsened space should have the same cost with respect to the objective function as its extension to the original space (i.e. the coarsening is exact): this requirement ensures that the coarsening is truly filtering the solution space. However, the paradigm does seem to work well even if this is not the case (and indeed, exact coarsening techniques are not always possible).

This still does not tell us *how* to coarsen a given problem and the examples in Sect. 3 suggest that it is very much problem-specific. However, it is often possible that construction heuristics (traditionally used to construct an initial feasible solution prior to single-level refinement) can be modified into coarsening heuristics (e.g. the TSP & VRP rely on this technique). Furthermore, it has been shown (for partitioning at least), that it is usually more profitable for the coarsening to respect the objective function in some sense (e.g. [21, 48]) and most construction heuristics (apart from completely randomised instances) have this attribute.

Multilevel Enhancements

As we saw in Sect. 3, there are a number of generic ideas that can be used to improve the performance of a multilevel algorithm (these are discussed in more

detail under multilevel enhancements in Sect. 3.1 and also in the summary of Sect. 3.2). In particular, techniques such as **constraint relaxation** and **coarsening homogeneity** seem to be beneficial and are worth investigation for other problems.

Perhaps the most powerful, however, is solution-based **recoarsening**, particularly if used as part of an iterated multilevel algorithm. This is usually very easy to implement (relying merely on the coarsening being capable of respecting an existing solution) and can often considerably improve the solution quality of a multilevel scheme (e.g. see Sect. 2.3).

```
iterated multilevel refinement( input problem instance P )
begin
    C ← multilevel refinement( P )   // initialise best solution
    i ← 0                            // unsuccessful iteration count
    while ( i < γ )                  // for intensity parameter γ
        C' ← multilevel refinement( P, C )
        if ( f(C') < f(C) )          // C' has lower cost
            C ← C'                   // update best solution
            i ← 0                    // reset unsuccessful count
        else
            i ← i+1
        endif
    end
end
```

Fig. 6. A schematic of the iterated multilevel refinement algorithm

Fig. 6 shows a schematic of a possible iterated multilevel algorithm (although there are other ways to specify the stopping criterion). Thus, after calculating an initial solution the algorithm repeatedly recoarsens and refines until no lower cost solution (where $f()$ denotes the objective/cost function) has been found after γ iterations. The only modification required to the multilevel algorithm of Fig. 5 is that it must take an existing solution as an additional input and coarsen that.

Summary

It seems likely that the most difficult aspect in implementing an effective multilevel scheme for a given problem and refinement algorithm is the (problem-specific) task of devising the coarsening strategy. However, examples indicate that this is often relatively straightforward.

4.3 Typical Runtime

One of the concerns that might be raised about multilevel algorithms is that instead of having just one problem to optimise, the scheme creates a whole hierarchy of approximately $O(\log_2 N)$ problems (assuming that the coarsening approximately halves the problem size at each level). In fact, it is not too difficult to show that there is approximately a factor-of-two difference in the runtime between a local search algorithm (or metaheuristic) at intensity λ, LS_λ, and the multilevel version, MLLS_λ. In other words, if $T(\mathrm{A})$ denotes the runtime of algorithm A, then $T(\mathrm{MLLS}_\lambda) \approx 2T(\mathrm{LS}_\lambda)$.

Suppose first of all that the LS algorithm is $O(N)$ in execution time and that the multilevel coarsening manages to halve the problem size at every step. This is an upper bound and in practice the coarsening rate is actually slightly lower (e.g. between 5/8 to 6/8 is typical for GPP and TSP examples [44], rather than the theoretic maximum of 1/2). Let $T_L = T(\mathrm{LS}_\lambda)$ be the time for LS_λ to run on a given instance of size N and T_C the time to coarsen and contract it. The assumption on the coarsening rate gives us a series of problems of size $N, N/2, N/4, \ldots, O(1)$ $(\approx N/N)$ whilst the assumption on LS_λ having linear runtime gives the total runtime for MLLS_λ as $T_C + T_L/N + \ldots + T_L/4 + T_L/2 + T_L$. If λ is small then T_C can take large proportion of the runtime and so multilevel algorithms using purely greedy refinement policies (i.e. typically $\lambda = 0$) tend to take more than twice the runtime of the equivalent local search although this depends on how long the coarsening takes compared with the solution construction (typically both $O(N)$). However, if λ is large enough then typically $T_C \ll T_L$ and so we can neglect it giving a total runtime of $T_L/N + \ldots + T_L/2 + T_L \approx 2T_L$, i.e. MLLS_λ takes twice as long as LS_λ to run.

Furthermore, if the local search scheme is also linear in λ, the intensity, it follows from $T(\mathrm{MLLS}_\lambda) \approx 2T(\mathrm{LS}_\lambda)$ that $T(\mathrm{MLLS}_\lambda) \approx T(\mathrm{LS}_{2\lambda})$. This effect is particularly visible for the TSP [42], but even for other problems where, for example λ expresses the number of failed iterations of some loop of the scheme, although this factor of two rule-of-thumb breaks down somewhat, it still gives a guide figure for the cost of a multilevel scheme.

Finally note that if the multilevel procedure is combined with an $O(N^2)$ or even $O(N^3)$ refinement algorithm then this analysis comes out even better for the multilevel overhead, i.e. $T(\mathrm{MLLS}_\lambda) < 2T(\mathrm{LS}_\lambda)$, as the final refinement step would requires an even larger proportion of the total.

Of course, this analysis assumes that the intensity of the search scheme is in some way dependent on the asymptotic convergence rate of the best solution. However, for local search schemes which are governed by a CPU time-limit, even tighter controls can be placed on the run-time of the multilevel version. Indeed, since the size of the problem at each level is known before any refinement takes place, the CPU time remaining after coarsening could be shared appropriately between each level and, for example, the multilevel version could be guaranteed to take the same time as the LS scheme.

5 Summary

We have seen evidence that the multilevel paradigm can aid metaheuristics and local search algorithms to find better or faster solutions for a growing number of combinatorial problems. We have discussed the generic features of these implementations and extracted some guidelines for its use. We have also identified some straightforward multilevel enhancements including recoarsening/iterated multilevel algorithms, homogeneous coarsening and constraint relaxation, all of which can boost its performance still further.

Overall this augments existing evidence that, although the multilevel framework cannot be considered as a panacea for combinatorial optimisation problems, it can provide a valuable (and sometimes a remarkably valuable) addition to the combinatorial optimisation toolkit. In addition, and unlike most metaheuristics, it is collaborative in nature and works alongside a separate refinement algorithm – successful examples include ant colony optimisation, cooperative search, genetic algorithms, simulated annealing and tabu search.

With regard to future work, clearly it is of great interest to further validate (or contradict) the conclusions by extending the range of problem classes. It is also valuable to identify and generalise those multilevel techniques which boost a particular application's performance still further (e.g. such as iterated multilevel schemes) so that they can become more widely employed.

References

1. A. Abou-Rjeili and G. Karypis. Multilevel algorithms for partitioning power-law graphs. In *Proc. 20th Intl Parallel & Distributed Processing Symp., 2006*, page 10 pp. IEEE, 2006.
2. S. T. Barnard and H. D. Simon. A Fast Multilevel Implementation of Recursive Spectral Bisection for Partitioning Unstructured Problems. *Concurrency: Practice & Experience*, 6(2):101–117, 1994.
3. R. Battiti, A. Bertossi, and A. Cappelletti. Multilevel Reactive Tabu Search for Graph Partitioning. Preprint UTM 554, Dip. Mat., Univ. Trento, Italy, 1999.
4. C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, 35(3):268–308, 2003.
5. C. Blum and M. Yábar. Multi-level ant colony optimization for DNA sequencing by hybridization. In F. Almeida *et al.*, editors, *Proc. 3rd Intl Workshop on Hybrid Metaheuristics*, volume 4030 of *LNCS*, pages 94–109. Springer, Berlin, Germany, 2006.
6. C. Blum, M. Yábar-Vallès, and M. J. Blesa. An ant colony optimization algorithm for DNA sequencing by hybridization. *Computers & Operations Research*, 2007. (in press).
7. E. G. Boman and B. Hendrickson. A Multilevel Algorithm for Reducing the Envelope of Sparse Matrices. Tech. Rep. 96-14, SCCM, Stanford Univ., CA, 1996.
8. N. Bouhmala. Combining local search and genetic algorithms with the multilevel paradigm for the traveling salesman problem. In C. Blum *et al.*, editors, *Proc. 1st Intl Workshop on Hybrid Metaheuristics*, 2004. ISBN 3-00-015331-4.

9. A. Brandt. Multiscale Scientific Computation: Review 2000. In T. J. Barth, T. Chan, and R. Haimes, editors, *Multiscale and Multiresolution Methods*, pages 3–95. Springer, Berlin, Germany, 2001.
10. T. N. Bui and C. Jones. A Heuristic for Reducing Fill-In in Sparse Matrix Factorization. In R. F. Sincovec *et al.*, editors, *Parallel Processing for Scientific Computing*, pages 445–452. SIAM, Philadelphia, 1993.
11. J. Cong and J. Shinnerl, editors. *Multilevel Optimization in VLSICAD*. Kluwer, Boston, 2003.
12. T. G. Crainic, Y. Li, and M. Toulouse. A First Multilevel Cooperative Algorithm for Capacitated Multicommodity Network Design. *Computers & Operations Research*, 33(9):2602–2622, 2006.
13. C. Dai, P. C. Li, and M. Toulouse. A Multilevel Cooperative Tabu Search Algorithm for the Covering Design Problem. Dept Computer Science, Univ. Manitoba, 2006.
14. C. M. Fiduccia and R. M. Mattheyses. A Linear Time Heuristic for Improving Network Partitions. In *Proc. 19th IEEE Design Automation Conf.*, pages 175–181. IEEE, Piscataway, NJ, 1982.
15. J. E. Gallardo, C. Cotta, and A. J. Fernández. A Multi-level Memetic/Exact Hybrid Algorithm for the Still Life Problem. In T. P. Runarsson *et al.*, editors, *Parallel Problem Solving from Nature – PPSN IX*, volume 4193 of *LNCS*, pages 212–221. Springer, Berlin, 2006.
16. J. Gu and X. Huang. Efficient Local Search With Search Space Smoothing: A Case Study of the Traveling Salesman Problem (TSP). *IEEE Transactions on Systems, Man & Cybernetics*, 24(5):728–735, 1994.
17. A. Gupta. Fast and effective algorithms for graph partitioning and sparse matrix reordering. *IBM Journal of Research & Development*, 41(1/2):171–183, 1996.
18. D. Harel and Y. Koren. A Fast Multi-Scale Algorithm for Drawing Large Graphs. *Journal of Graph Algorithms & Applications*, 6(3):179–202, 2002.
19. B. Hendrickson and R. Leland. A Multilevel Algorithm for Partitioning Graphs. In S. Karin, editor, *Proc. Supercomputing '95, San Diego (CD-ROM)*. ACM Press, New York, 1995.
20. Y. F. Hu and J. A. Scott. Multilevel Algorithms for Wavefront Reduction. RAL-TR-2000-031, Comput. Sci. & Engrg. Dept., Rutherford Appleton Lab., Didcot, UK, 2000.
21. G. Karypis and V. Kumar. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.
22. G. Karypis and V. Kumar. Multilevel Algorithms for Multi-Constraint Graph Partitioning. In D. Duke, editor, *Proc. Supercomputing '98, Orlando*. ACM SIGARCH & IEEE Comput. Soc., 1998. (CD-ROM).
23. G. Karypis and V. Kumar. Multilevel k-way Partitioning Scheme for Irregular Graphs. *Journal of Parallel & Distributed Computing*, 48(1):96–129, 1998.
24. G. Karypis and V. Kumar. Multilevel k-way Hypergraph Partitioning. *VLSI Design*, 11(3):285–300, 2000.
25. A. Kaveh and H. A. Rahimi-Bondarabady. A Hybrid Graph-Genetic Method for Domain Decomposition. In B. H. V. Topping, editor, *Computational Engineering using Metaphors from Nature*, pages 127–134. Civil-Comp Press, Edinburgh, 2000. (Proc. Engrg. Comput. Technology, Leuven, Belgium, 2000).
26. B. W. Kernighan and S. Lin. An Efficient Heuristic for Partitioning Graphs. *Bell Systems Technical Journal*, 49:291–308, 1970.

27. P. Korošec, J. Šilc, and B. Robič. Solving the mesh-partitioning problem with an ant-colony algorithm. *Parallel Computing*, 30:785–801, 2004.
28. A. E. Langham and P. W. Grant. A Multilevel k-way Partitioning Algorithm for Finite Element Meshes using Competing Ant Colonies. In W. Banzhaf *et al.*, editors, *Proc. Genetic & Evolutionary Comput. Conf. (GECCO-1999)*, pages 1602–1608. Morgan Kaufmann, San Francisco, 1999.
29. I. O. Oduntan. A Multilevel Search Algorithm for Feature Selection in Biomedical Data. Master's thesis, Dept. Computer Science, Univ. Manitoba, 2005.
30. D. Rodney, A. Soper, and C. Walshaw. The Application of Multilevel Refinement to the Vehicle Routing Problem. In D. Fogel *et al.*, editors, *Proc. CISChed 2007, IEEE Symposium on Computational Intelligence in Scheduling*, pages 212–219. IEEE, Piscataway, NJ, 2007.
31. D. F. Rogers, R. D. Plante, R. T. Wong, and J. R. Evans. Aggregation and Disaggregation Techniques and Methodology in Optimization. *Operations Research*, 39(4):553–582, 1991.
32. Y. Romem, L. Rudolph, and J. Stein. Adapting Multilevel Simulated Annealing for Mapping Dynamic Irregular Problems. In S. Ranka, editor, *Proc. Intl Parallel Processing Symp.*, pages 65–72, 1995.
33. Camilo Rostoker and Chris Dabrowski. Multilevel Stochastic Local Search for SAT. Dept Computer Science, Univ. British Columbia, 2005.
34. Y. Saab. Combinatorial Optimization by Dynamic Contraction. *Journal of Heuristics*, 3(3):207–224, 1997.
35. I. Safro, D. Ron, and A. Brandt. Graph minimum linear arrangement by multilevel weighted edge contractions. *Journal of Algorithms*, 60(1):24–41, 2006.
36. K. Schloegel, G. Karypis, and V. Kumar. Multilevel Diffusion Schemes for Repartitioning of Adaptive Meshes. *Journal of Parallel & Distributed Computing*, 47(2):109–124, 1997.
37. K. Schloegel, G. Karypis, and V. Kumar. A New Algorithm for Multi-objective Graph Partitioning. In P. Amestoy *et al.*, editors, *Proc. Euro-Par '99 Parallel Processing*, volume 1685 of *LNCS*, pages 322–331. Springer, Heidelberg, Germany, 1999.
38. A. J. Soper, C. Walshaw, and M. Cross. A Combined Evolutionary Search and Multilevel Optimisation Approach to Graph Partitioning. *Journal of Global Optimization*, 29(2):225–241, 2004.
39. S.-H. Teng. Coarsening, Sampling, and Smoothing: Elements of the Multilevel Method. In M. T. Heath *et al.*, editors, *Algorithms for Parallel Processing*, volume 105 of *IMA Volumes in Mathematics and its Applications*, pages 247–276. Springer, New York, 1999.
40. M. Toulouse, K. Thulasiraman, and F. Glover. Multi-level Cooperative Search: A New Paradigm for Combinatorial Optimization and an Application to Graph Partitioning. In P. Amestoy *et al.*, editors, *Proc. Euro-Par '99 Parallel Processing*, volume 1685 of *LNCS*, pages 533–542. Springer, Berlin, 1999.
41. D. Vanderstraeten, C. Farhat, P. S. Chen, R. Keunings, and O. Zone. A Retrofit Based Methodology for the Fast Generation and Optimization of Large-Scale Mesh Partitions: Beyond the Minimum Interface Size Criterion. *Computer Methods in Applied Mechanics & Engineering*, 133:25–45, 1996.
42. C. Walshaw. A Multilevel Approach to the Travelling Salesman Problem. *Operations Research*, 50(5):862–877, 2002.
43. C. Walshaw. A Multilevel Algorithm for Force-Directed Graph Drawing. *Journal of Graph Algorithms & Applications*, 7(3):253–285, 2003.

44. C. Walshaw. Multilevel Refinement for Combinatorial Optimisation Problems. *Annals of Operations Research*, 131:325–372, 2004.
45. C. Walshaw. Variable partition inertia: graph repartitioning and load-balancing for adaptive meshes. In S. Chandra M. Parashar and X. Li, editors, *Advanced Computational Infrastructures for Parallel and Distributed Adaptive Applications*. Wiley, New York, 2007. (Invited chapter – in press).
46. C. Walshaw and M. Cross. Mesh Partitioning: a Multilevel Balancing and Refinement Algorithm. *SIAM Journal on Scientific Computing*, 22(1):63–80, 2000.
47. C. Walshaw and M. Cross. Multilevel Mesh Partitioning for Heterogeneous Communication Networks. *Future Generation Computer Systems*, 17(5):601–623, 2001.
48. C. Walshaw, M. Cross, R. Diekmann, and F. Schlimbach. Multilevel Mesh Partitioning for Optimising Domain Shape. *International Journal of High Performance Computing Applications*, 13(4):334–353, 1999.
49. C. Walshaw and M. G. Everett. Multilevel Landscapes in Combinatorial Optimisation. Tech. Rep. 02/IM/93, Comp. Math. Sci., Univ. Greenwich, London SE10 9LS, UK, April 2002.