Linear Linkage Encoding in Grouping Problems: Applications on Graph Coloring and Timetabling

Özgür Ülker, Ender Özcan and Emin Erkan Korkmaz

Department of Computer Engineering
Yeditepe University
Istanbul, TURKEY
{oulker, eozcan, ekorkmaz}@cse.yeditepe.edu.tr
http://cse.yeditepe.edu.tr/ARTI

Abstract. Linear Linkage Encoding (LLE) is a recently proposed representation scheme for evolutionary algorithms. This representation has been used only in data clustering. However, it is also suitable for grouping problems. In this paper, we investigate LLE on two grouping problems; graph coloring and final exam timetabling. Two crossover operators suitable for LLE are proposed and compared to the existing ones. Initial results show that Linear Linkage Encoding is a viable candidate for grouping problems whenever appropriate genetic operators are used.

1 Introduction

In spite of the satisfactory performance of Evolutionary Algorithms (EA) on many NP Optimization problems, the same achievement is not usually observed on grouping problems where the task is to partition a set of objects into disjoint sets. This is because the commonly used representations usually suffer from redundancies due to the ordering of groups. Moreover the genetic material might easily be disrupted by the genetic operators and/or by the rectification process after the operators are applied.

Timetabling problems are real world NP Hard [7] problems. Discarding the rest of the constraints, attempting to minimize the timetabling slots while satisfying the clashing constraints turn out to be graph coloring problem [19]. For this reason, new representation schemes and operators used in graph coloring are also of interest to the researchers in timetabling community.

In the paper, we are investigating a recently proposed encoding scheme for grouping problems, Linear Linkage Encoding (LLE) [6]. LLE has only been tested on small clustering problem instances, and authors claim that the LLE performance is superior to Number Encoding (NE), the most common encoding scheme used in grouping problems. Unlike NE, LLE does not require an explicit bound on the number of groups that can be represented in a fixed-length chromosome. The greatest strength of LLE is that the search space is reduced considerably. There is a one to one correspondence between the chromosomes and the solutions when LLE is used. Consequently the aim of this paper is to present the potential of the LLE representation on grouping problems. Previous studies denote that traditional crossover operators do not perform well. Therefore,

a set of new crossover operators suitable for LLE are also tested on a set of problem instances including Carter's Benchmark [5] and DIMACS Challenge Suite [18].

This paper is organized as follows: We first define the grouping problems and common representations for them. The fundamentals of Linear Linkage Encoding is followed by the definition of the graph coloring problem. Then the operators of the algorithm with special crossovers are presented. Computational experiments and conclusions are given at the end of the paper.

2 Grouping Problems

Grouping problems [8] are generally concerned with partitioning a set V of items into a collection of mutually disjoint subsets V_i of V such that

$$V = V_1 \cup V_2 \cup V_3 ..... \cup V_N \text{ and } V_i \cap V_j = \emptyset \text{ where } i \neq j.$$

Obviously, the aim of these problems is to partition the members of set *V* into *N* different groups where $(1 \leq N \leq |V|)$ each item is in exactly one group. In most of the grouping problems, not all possible groupings are permitted; a valid solution usually has to comply a set of constraints. For example in graph coloring, the vertices in the same group must not be adjacent in the graph. In bin packing problem, sum of the sizes of items of any group should not exceed the capacity of the bin, etc. Hence, the objective of grouping is to optimize a cost function defined over a set of valid groupings. In both graph coloring and bin packing the objective is to minimize the number of groups (independent sets and bins respectively) subjected to the mentioned constraints.

Grouping problems are characterized by the cost function based on the composition of the groups. An item in isolation has little or no meaning during the search process. Therefore, the building blocks that should be preserved in an evolutionary search should be the groups or the group segments.

2.1 Representations in Grouping Problems

The most predominant representation in grouping problems in both evolutionary and local search methods is Number Encoding (NE). In NE, each object is encoded with a group id indicating which group it belongs to. For example the individual 2342123 encodes the solution where first object is in group 2, second in 3, third in 4, and so on. However, it is easy to see that the encoding 1231412 represents exactly the same solution, since the naming or the ordering of the partitions is irrelevant. The drawbacks of this representation are presented in [8] and it is pointed out that this encoding is against the minimal redundancy principles for encoding scheme [24].

Another representation for grouping problems is Group Encoding (GE). The objects which are in the same group are placed into the same partition. For instance, the above sequence can be represented as $(1, 4, 6)(2, 7)(3)(5)$. The ordering within each partition is unimportant, since search operators work on groups rather than objects unlike in NE. However the ordering redundancy among groups still holds. For instance, $(2, 7)(3)(5)(1, 4, 6)$ would again represent the same solution.

2.2 Linear Linkage Encoding

LLE can be implemented using an array. Let the entries in the chromosome be indexed with values from 1 to *n*. Each entry in the array then holds one integer value which is a link from one object to another object of the same partition. With *n* objects, any partition on them can be represented as an array of length *n*. Two objects are in the same partition if either one can be reached from another through the links. If an entry is equal to its own index, then it is considered as an ending node. The links in LLE are unidirectional, thus; backward links are not allowed. In short, in order to be considered as a valid LLE array, the chromosome should follow the following two rules:

- The integer value in each entry is greater than or equal to its index but less than or equal to *n*.
- No two entries in the array can have the same value; the index of an ending node is the only exception to this rule.

In LLE, the items in a group construct a linear path ending with a self referencing last item. It can be represented by the *labeled oriented pseudo (LOP)* graph. A LOP Graph is a labeled directed graph $G(V, E)$, where V is the vertex set and E is the edge set. A composition of G is a grouping of $V(G)$ into disjointed oriented pseudo path graphs $G_1, G_2, ...G_m$ with the following properties:

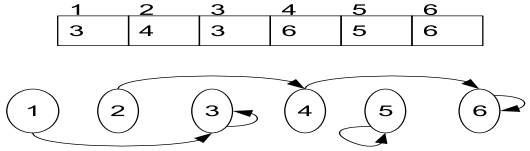

Fig. 1. LLE Array and LOP Graphs

- Disjoint paths: $\bigcup_{i=1}^{m} V(G_i) = V(G)$ and for $i \neq j, V(G_i) \bigcap V(G_j) = \emptyset$
- Non-backward oriented edges: If there is an edge *e* directed from vertex v_i to v_k then $i \leq k$.
- Balanced Connectivity
  - a. $|E(G)| = |V(G)|$
  - b. each G_i has only one ending node with an *in-degree* of 2 and *out-degree* of 1.
  - c. each G_i has only one starting node whose *in-degree* $= 0$ and *out degree* $= 1$
- All other $|V(G_i)| - 2$ vertices in G_i have *in-degree* $=$ *out-degree* $= 1$.

There are three clear observations regarding LOP Graphs:

1. Given a set of items S, there is one and only one composition of LOP Graphs $G(V, E)$ for each grouping of S, where $|V| = |S|$.
2. The number of LOP Graphs is given by the n^{th} Bell Number [6].
3. LLE in array form is a unique implementation of the LOP graph.

2.3 Final Exam Timetabling as a Grouping Problem

Final exam timetabling requires satisfactory assignment of timetable slots (periods) to a set of exams. Each exam is taken by a number of students, based on a set of constraints. In most of the studies, NE like representations are used. In [3], a randomly selected light or a heavy mutation followed by a hill climbing method was applied. Various combinations of constraint satisfaction techniques with genetic algorithms can be found in [20]. Paquete et. al. [23] applied a multi-objective evolutionary algorithm based on pareto ranking with two objectives: minimize the number of conflicts within the same group and between groups. Wong et. al [26] applied a GA with a non-elitist replacement strategy. After genetic operators are applied, violations are repaired with a hill climbing fixing process. In their experiments a single problem instance was used. Ozcan et. al. [22] proposed a memetic algorithm (MA) for solving final exam timetabling at Yeditepe University. MA utilizes a violation directed adaptive hill climber.

Considering the task of minimizing the number of exam periods and removing the clashes, final exam timetabling reduces to the graph coloring problem [19].

2.4 Graph Coloring Problem as a Grouping Problem

Graph Coloring (GCP) is a well known combinatorial optimization problem which is proved to be NP Complete [11]. Informally stated, graph coloring is assigning colors to each vertex of an undirected graph such that no adjacent vertices should receive the same color. The minimal number of colors that can be used for a valid coloring is called the chromatic number. A more formal definition is as follows:

Given a graph $G = (V, E)$ with vertex set V and edge set E, and given an integer *k*, a *k-coloring* of *G* is a function $c : V \rightarrow 1, ..., k$. The value $c(x)$ of a vertex *x* is called the color of *x*. The vertices with color *r* $(1 \leq r \leq k)$ define a color class, denoted V_r. If two adjacent vertices *x* and *y* have the same color *r*, *x* and *y* are conflicting vertices, and the edge (x, y) is called a conflicting edge. If there is no conflicting edge, then the color classes are all independent sets and the *k-coloring* is valid. The Graph Coloring Problem is to determine the minimum integer *k* (the chromatic number of *G* - $\chi(G)$) such that there exists a legal *k-coloring* of *G* [1].

In the literature there are many heuristics devised for finding chromatic number and solving k-coloring problems. Early applications of GCP solvers are simple constructive methods [2], [19] which color each vertex of the graph one after another based on dynamic ordering of the vertices according to its saturation degree as in DSATUR. Local search methods such as tabu search [14] and simulated annealing [16] have been followed with hybridizations of these techniques with genetic algorithms [9], [10] which resulted the state of the art graph coloring algorithms.

Graph coloring is generally considered as a difficult problem for pure Genetic Algorithms [13]. Currently, the most successful algorithms are memetic algorithms [9], [10] which hybridize the evolutionary techniques with a local search method. In this approach, the role of genetic operators is limited to finding promising points in the search landscape from which the local search can initiate. Hence, the exploration of the search space is carried out by the local search operator. For instance in Galinier and Hao's

hybrid algorithm [10], a crossover operation is proceeded by a tabu search procedure which may last thousands of tabu iterations.

There are mainly two reasons for the unsuccessful attempts of using pure genetic implementations on graph coloring: The redundancies inherent in the representations used for the encoding of the chromosome, and lack of a suitable crossover operator which would transmit the building blocks efficiently, preferably with some domain knowledge. In this paper, we are mainly interested in the representational issues, but we also present suitable crossover operators for the proposed multi-objective genetic algorithm.

3 A Multi-Objective Genetic Algorithm for Graph Coloring and Timetabling

Note that our main intention in this study is to propose a multi objective solution foundation to multi-constraint timetabling problems. To our knowledge, none of the efficient graph coloring algorithms in the literature empowers genetic operators as their main search mechanism. These methods usually rely on local search operators. We are more interested in the applicability of linear linkage encoding on grouping problems by using suitable crossover and mutation operators. We present a multi-objective genetic algorithm employing weak elitism and the main search operator of this approach is mutation aided by crossover.

3.1 Initialization

Since we are dealing with a minimal coloring problem, it is desirable to initialize the population with individuals having different number of colors. Setting the range of number of colors too wide will unnecessarily increase the search space and thus the execution time. For example, in a graph instance with 128 vertices and with a predicted chromatic number of 20, setting the range between the minimum and maximum values (*1* and the $|V|$ which is *128* in this case) will force the algorithm to survey individuals far away from the optimal coloring. It is also undesirable to set the range too narrow either. Such a scheme will prevent promising individuals with different number of colors from cooperating through crossover and mutation.

A tight lower bound can be found by using a fast maximal clique approximation algorithm and an upper bound can be found by simply finding the maximal degree of the graph (the degree of the vertex with the maximum number of neighbors). Since exact or approximate chromatic numbers in the test instances are already known, these bounds are set manually in this study.

In our experiments, we have used a population with individuals having different number of colors and an external population which holds the best individuals with the minimal conflicts for a specific number of colors within a search range ($lowerBound \leq k \leq upperBound$). In order to create an individual, first *k* is determined, then a k-colored individual is randomly created. An external smart initialization method was not used to reduce the edge conflicts in order not to give any bias to our crossover operators and let the multi-objective evolutionary method do the search.

3.2 Selection

A k-coloring problem is solved when the number of conflicting edges is zero. If a *k coloring* solution is obtained, $k+1$ colorings can also be generated by dividing independent sets into two. It might be possible to unite two sets in a *k+1 coloring* to obtain a *k coloring*. Therefore, if the chromatic number lies within the upper and lower bounds, the pareto front will almost be a straight line along the color axis with zero conflict if the lower bound is set close to the chromatic number. A restricted multi-objective method might work efficiently on a search range within specified bounds around the chromatic number.

As a multi-objective genetic algorithm a modified version of Niched Pareto Genetic Algorithm (NPGA) described in [15] was used. In NPGA, two candidate individuals are selected at random from the population to be one of the mates. A comparison set is formed from randomly selected individuals within the population. Each candidate is then compared against each individual in the comparison set. If one candidate is dominated by the comparison set (which means it is worse for every part of the objective function than any individual in the comparison set) and the other is not, then the latter is selected for reproduction. If neither or both are dominated by the comparison set, then niching is used to select a winner mate. The size of comparison set (t_{dom}) allows a control over the selection pressure. The comparison set size was preset to around ten percent of the population size as suggested by [15].

When neither or both candidates are dominated by the comparison set, the candidate with a smaller niche count is selected for reproduction. We calculate the niche value m_i of the i^{th} individual by:

$$m_i = \sum_{j \in pop} sh(d[i,j]) \tag{1}$$

where *d[i,j]* is the distance between two individuals according to objective function values and *sh(d)* is the sharing function which is:

$$sh(d) = \begin{cases} 1 & \text{if } d = 0 \\ 1 - d/\mu_{share} & \text{if } d < \mu_{share} \\ 0 & \text{if } d \geq \mu_{share}. \end{cases} \tag{2}$$

and distance measure is Manhattan distance in terms of color and conflict values in the individuals.

$$d[i,j] = |c_{i1} - c_{j1}| + |c_{i2} - c_{j2}| \tag{3}$$

3.3 Redundancy and Genetic Operators

Although LLE in theory is a non-redundant representation for grouping problems, practically this advantage disappears if the search operators do not adhere to this principle. Therefore a more desirable option is to make the search non-redundant additional to the representation. For example consider a basic hill climbing mutation which sends one vertex from one set to another. This is analogous of changing a gene value in the number encoding. If majority of the group ids of the items can be maintained for a long period of time, then it is quite possible to make a low-redundant search even on a highly

redundant encoding such as NE. This is one of the reasons local search based methods are quite successful on grouping problems. Because of the small perturbations on the search space, these methods not only preserve the building blocks on the candidate solution but also are able to operate on a low-redundant small region of the large search landscape.

The same advantage, unfortunately does not hold for crossover which makes huge jumps on the search space. It is possible to keep the majority of the group ids of the items fixed by using traditional crossovers like one-point or uniform crossover. Such methods, however do not preserve the groups which are the building blocks themselves. Therefore, a crossover operator should preserve the order of the colors as long as possible. Two ordering mechanism which assigns group ids to the groups after crossover and mutation are investigated within the context of LLE. These two redundancy elimination mechanisms are based on the cardinality of the groups and the lowest index number at each group. In [25], the authors investigated the effect of these two methods on Graph Coloring by using 0/1 ILP SAT solvers.

Cardinality Based Ordering In Cardinality Based Ordering, each group receives a group id according to its cardinality (set size). Groups are sorted according to their cardinality and the group with the highest cardinality will be assigned group id 1, the second highest will be identified as group 2, and so on. For example groups $(1,3)(5)(2,4,6)$ are indexed as $V_1 = (2,4,6)$, $V_2 = (1,3)$, and $V_3 = (5)$. Since more than one group can have the same cardinality, the ordering might not be unique.

Lowest Index Ordering In Lowest Index Ordering, the smallest index in each group is found first, then the group with the smallest index number is assigned group id 1, the group with the second smallest index number is assigned group id 2, and so on. For example, groups $(1,3)(5)(2,4,6)$ are indexed as $V_1 = (1,3)$, $V_2 = (2,4,6)$, and $V_3 = (5)$. Since each group has one unique lowest index, the ordering is always unique.

3.4 Crossover

Linear linkage encoding can be implemented using one dimensional arrays, allowing applicability of the traditional crossover methods such as, one point or uniform crossover. However, it is observed that these crossovers can be too destructive especially for graph coloring due to the danger of introducing new links in the LOP graph absent in both parents. Also since the building blocks [12] in graph coloring are strictly large independent sets (not even independent set segments), there is a risk of destructing these building blocks. However, for small problem instances, one-point crossover in LLE is reported to generate satisfactory results for clustering problem [6]. (This might be due to the fact that building blocks may be a segment of clusters rather than the whole cluster.)

Unfortunately we have observed a very poor performance from one point crossover in our experiments. It was not even able to generate solutions in the color search range we specified.

Three types of crossover operators are compared using LLE representation.

Greedy Partition Crossover Graph Coloring Problem can be considered as partitioning the graph into independent sets. Therefore by preserving the large independent sets, the vertices in non-independent sets can be forced to form independent sets as well.

Greedy Partition Crossover (GPX) was proposed by Galinier and Hao [10] in their Hybrid Graph Coloring Algorithm. The idea is to transmit the largest set (group) from one parent, then to delete the vertices in this largest set from the other parent. This transmission and deletion process is repeated on both parents successively until all of the vertices are assigned to the child.

Two forms of Greedy Partition Crossover by following the rules of Cardinality and Lowest Index Ordering are implemented. The difference is just assigning the color ids to the groups after the crossover. In GPX Lowest Index Crossover (GPX-LI), the groups with lower index numbers are given lower color ids, whereas in GPX Cardinality Based Crossover (GPX-CB), the lower color ids are assigned to the groups with higher cardinality. A general pseudocode of GPX is presented in Algorithm 1.

Consider two parents in Figure 2. We can obtain the child as follows: Largest Set in parent 1 is $(3, 4, 5, 6)$. This set is transmitted to the child and 1, 3 and 6 are deleted from parent 2. After this deletion largest set in parent 2 (1) is transmitted to the child. Finally (2) is assigned as the last group. After sorting according to lowest index ordering (GPX-LI), the coloring then becomes $C_1 = (1)$, $C_2 = (2)$, $C_3 = (3, 4, 5, 6)$. If the groups are sorted according to their cardinality (GPX-CB), the coloring is $C_1 = (3, 4, 5, 6)$, $C_2 = (1)$, $C_3 = (2)$.

Both GPX-LI and GPX-CB are applicable to other representations such as number or group encodings. Our intention of using these crossovers is to create crossover operators applicable only to LLE. The following two crossovers are inspired from GPX.

Algorithm 1 Greedy Partition Crossover

Require: Two Parents - $parent1$ and $parent2$ in LLE form.
Ensure: One $offspring$ in LLE form.

repeat
  $currentParent$ = Random($parent1$, $parent2$).
  $largestSet$ = Find largest set in $currentParent$.
  transmit unassigned the vertices (links) in the $largestSet$ to $offspring$.
  mark transmitted vertices as assigned.
  if $currentParent = parent1$ **then**
    $currentParent$ = $parent2$.
  else
    $currentParent$ = $parent1$.
  end if
until all vertices are assigned
if Lowest Index Ordering is Used **then**
  sort group ids according to lowest index number (GPX-LI).
else
  sort group ids according to cardinality (GPX-CB).
end if

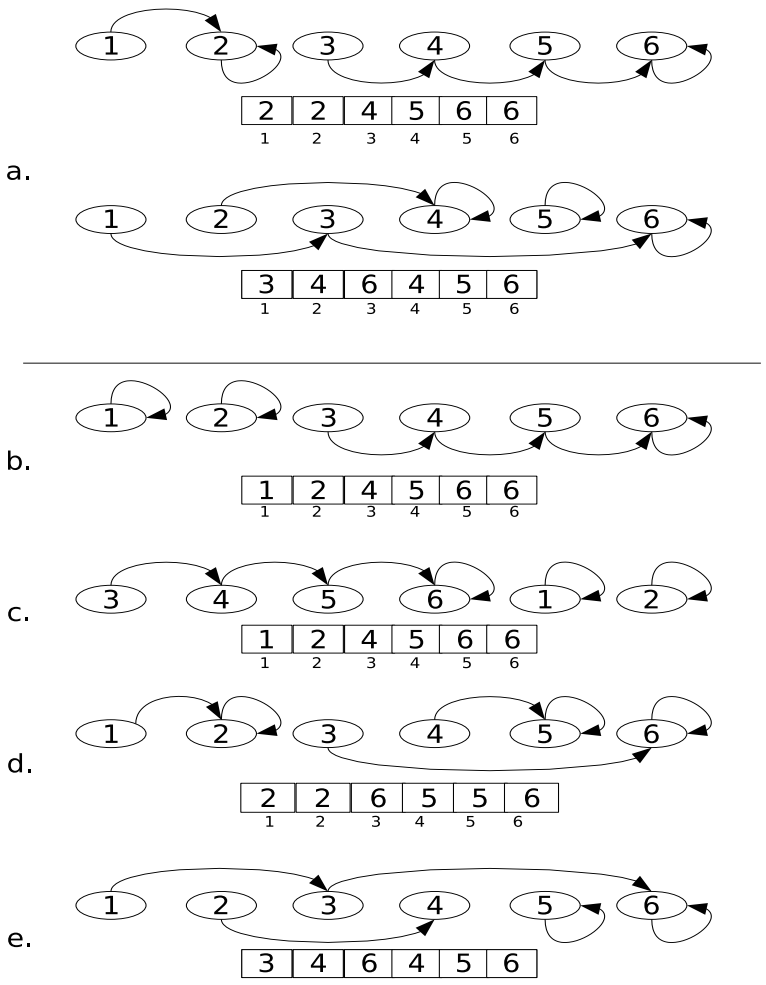

Fig. 2. a) Two Parents in LLE Array and LOP Graph form. b) Resulting offspring from Greedy Partition Crossover - Cardinality Based Ordering. c) Resulting offspring from Greedy Partition Crossover - Lowest Index Ordering. d) Resulting offspring from Lowest Index First Crossover. e) Resulting offspring from Lowest Index Max Crossover.

Lowest Index First Crossover In Lowest Index First Crossover (LIFX), the goal is to transmit the groups beginning with lowest index numbers. LIFX works as follows:

A parent is randomly selected. Beginning with the lowest index (vertex) which has not been assigned yet, the vertices are transmitted to the child by following the links. If the vertices along the path are assigned before, they are skipped. The process is repeated by successively changing the parents for transmission until all of the vertices are assigned to the child. A general pseudocode of LIFX is presented in Algorithm 2.

The application of LIFX on the parents in Figure 2 would be as follows: Assuming we begin with first parent, current lowest index number is 1. Therefore, $(1, 2)$ is transmitted to the child. The current lowest index number is now 3. Switching to parent 2, we copy $(3, 6)$ as the next group. Switching back to parent 1, current lowest index is 4, therefore $(4, 5)$ is copied to the child. Final coloring then becomes: $C_1 = (1, 2)$, $C_2 = (3, 6)$, $C_3 = (4, 5)$.

Note that this crossover prioritizes groups beginning with the lowest index number, therefore it reduces the sizes of the groups beginning with higher index numbers. This is in concordance with the nature of LLE, because the number of possible values for the higher index locations is lower.

Algorithm 2 Lowest Index First Crossover

Require: Two Parents - $parent1$ and $parent2$ in LLE form.
Ensure: One $offspring$ in LLE form.
1: $i = 0$
2: **repeat**
3: $currentParent$ = Random($parent1$, $parent2$).
4: $lengthOfParent$ = Calculate the path length of $currentParent$ starting from i.
5: transmit unassigned vertices (links) in the $parentToSelect$ to $offspring$.
6: mark transmitted vertices as assigned.
7: i = next unassigned vertex.
8: **if** $currentParent = parent1$ **then**
9: $currentParent$ = $parent2$.
10: **else**
11: $currentParent$ = $parent1$.
12: **end if**
13: **until** all vertices are assigned

Lowest Index Max Crossover In Lowest Index Max Crossover (LIMX), the child is generated with two objectives: Transmit large groups to preserve Cardinality Based Ordering, and to transmit groups beginning with lowest index number (to preserve Lowest Index Ordering). Therefore this method can be considered as an amalgamate of LIFX and GPX. LIMX works as follows:

Beginning with the lowest index number (vertex) which has not been assigned first we calculate the length of the links (path length) in both parents. Already assigned vertices are not counted in this link length calculation. This allows finding the largest

set in parents beginning with the lowest index number. Then the links (and thus vertices) are transmitted to the child from the parent with the greater link-length. After that next unassigned lowest index number is found and the process is repeated until all vertices are assigned. A general pseudocode of LIMX is presented in Algorithm 3.

Application of LIMX to parents in Figure 2 is as follows: Current lowest index is 1. $(1, 3, 6)$ is longer than $(1, 2)$ so $(1, 3, 6)$ is copied to the child. Current lowest index is now 2. $(2, 4)$ is larger than (2) so it is tranmitted to the child. Finally (5) is copied to the child as the last group. At the end of LIMX the coloring then becomes: $C_1 = (1, 3, 6)$, $C_2 = (2, 4)$, $C_3 = (5)$

Algorithm 3 Lowest Index Max Crossover

Require: Two Parents - $parent1$ and $parent2$ in LLE form.
Ensure: One $offspring$ in LLE form.
1: $i = 0$
2: **repeat**
3: $lengthOfParent1$ = Calculate the path length of $parent1$ starting from i.
4: $lengthOfParent2$ = Calculate the path length of $Parent1$ starting from i.
5: **if** $LengthOfParent1 < LengthOfParent2$ **then**
6: $parentToSelect$ = $parent1$.
7: **else**
8: $parentToSelect$ = $parent2$.
9: **end if**
10: transmit unassigned vertices (links) in the $parentToSelect$ to $offspring$.
11: mark transmitted vertices as assigned.
12: i = next unassigned vertex.
13: **until** all vertices are assigned

3.5 Mutation

We have used a mutation scheme that sends a selected conflicting vertex *x* from its color set to the best possible other one. A tournament method is used to select a vertex for transfer. A percentage of conflicting vertices are taken into a tournament and the vertex with the highest conflict in this set is transferred to a best color available.

This mutation can increase or decrease the number of colors by one. The number of colors will increase if a one vertex group is introduced by the operator and it will decrease if a one vertex group is merged with another one.

As aforementioned, assigning group ids after crossover is essential for low redundancy and the success of the mutation. In GPX-LI, LIMX and LIFX, the ids are assigned according to Lowest Index Ordering whereas in GPX-CB the ids are assigned according to Cardinality Based Ordering.

3.6 Replacement

In our simulations we have employed a trans-generational replacement with weak elitism. At each generation, λ offspring are produced by λ (non elitist)+ μ (elitist individuals,

one for each number of colors within the searching range) parents to replace the λ non elitist parents. By using elitism to hold the best conflict per number of colors we are giving a bias to number of colors constraint.

4 Experiments

In our tests, we use several graphs from the DIMACS Challenge Suite [18]. The general test setup is summarized in Table 1.

Table 1. Test Setup

Test Machine:	Pentium 4 2Ghz with 256MB Ram
Compiler:	GCC C++ 3.2 with -O2 flags
No of Generations:	10000
Population Size:	%25 percent of the number of vertices in graph
Comparison Set Size:	%10 percent of the population size
Niche Size:	5.0
Crossover Rate:	0.25
Mutation Rate:	a single mutation is enforced
Number of Runs:	50 for each instance

In Table 2, we present the characteristics of the test instances sampled from the DIMACS test suite. Table shows the name, number of vertices($|V|$), number of edges ($|E|$), edge density ($\%$) and chromatic number ($\chi(G)$) of the instances.

In all our tests, the mutation count is set to 1, and crossover rate is fixed at 0.25. In this setup, the algorithm is more like a genetic hill climbing method. Since the chromatic number of these graphs are already known, we have set the range by hand according to the chromatic number $\chi(G)$.

Note that our primary intention is to compare the crossover operators in the context of LLE. As a result, we did not run our experiments for a long time. (The longest time required for one run is around 5 minutes for cars91 graph instance). This might have resulted in performance hit for large problem instances which may need an exponential increase rather than a linear increase in the maximum number of generation.

In Table 3, we present the best solutions obtained after 50 runs by using the four crossover operators mentioned. Figure 3 represents the average color number of 50 runs for each instance. The results show no significant statistical differences between crossover operators except for a few instances. For example for flat300_20 graph, LIMX was able to find a best 20 coloring while the other crossovers were very far from the optimal. However, for this graph, average colorings found with all crossovers and standard deviation are quite high. This is possibly due to the natural difficulty of flat graphs. Another slight difference appeared in register allocation graphs (zeroin.X.col graphs) where LIFX performed worst while GPX crossovers performed best.

We have also presented graph coloring algorithm results of Kirovski et. al. [17] for two set of parameters (Kirovski B and Kirovski C). Kirovski's algorithm is based on

Table 2. Data Characteristics about the problem instances from the DIMACS Suite

Instance	$\|V\|$	$\|E\|$	%	$\chi(G)$
DSJC125.5	125	3891	0,50	?
DSJC125.9	125	6961	0,90	?
zeroin.1.col	211	4100	0,19	49
zeroin.2.col	211	3541	0,16	30
zeroin.3.col	206	3540	0,17	30
DSJC250.1	250	3218	0,10	?
DSJC250.5	250	15668	0,50	?
DSJC250.9	250	27897	0,90	?
flat300_20	300	21375	0,48	20
flat300_26	300	21633	0,48	26
flat300_28	300	21695	0,48	28
school1_nsh	352	14612	0,24	14
le450_15a	450	8168	0,08	15
le450_15b	450	8169	0,08	15
le450_15c	450	16680	0,17	15
le450_15d	450	16750	0,17	15
le450_25a	450	8260	0,08	25
le450_25b	450	8263	0,08	25
le450_25c	450	16680	0,17	25
le450_25d	450	16750	0,17	25
DSJC500.1	500	12458	0,10	?
DSJC500.5	500	62624	0,50	?

divide and conquer paradigms, global search for constrained independent sets, assignment of most-constrained vertices to least constraining colors,reuse and locality exploration of intermediate solutions, post processing lottery-scheduling iterative improvement. With respect to Kirovski's solutions, our crossovers gave similar and for some instances better results however when the instance becomes larger and more difficult, Kirovski's algorithm performs better. However, our primary intention was not to compare LLE representation with state of the art algorithms but to compare the crossover operators as stated before.

Table 3. Best colorings obtained for the instances in the DIMACS Benchmark Suite

Instance	$\chi(G)$	LIMX	LIFX	GPX-LI	GPX-CB	Kirovski-B	Kirovski-C
DSJC125.5	?	18	18	18	18	19	18
DSJC125.9	?	44	44	44	44	45	45
zeroin.1.col	49	49	50	49	49	49	49
zeroin.2.col	30	31	35	31	31	30	30
zeroin.3.col	30	31	35	30	31	30	30
DSJC250.1	?	9	9	9	9	9	9
DSJC250.5	?	31	31	31	31	30	30
DSJC250.9	?	75	75	75	74	77	77
flat300_20	20	20	31	27	32	20	20
flat300_26	26	34	34	34	34	32	28
flat300_28	28	34	34	34	34	33	32
school1_nsh	14	14	14	14	14	16	14
le450_15a	15	16	16	16	16	17	17
le450_15b	15	16	16	16	16	17	17
le450_15c	15	23	23	23	23	22	21
le450_15d	15	23	23	23	23	22	21
le450_25a	25	25	25	25	25	25	25
le450_25b	25	25	25	25	25	25	25
le450_25c	25	28	29	28	28	28	28
le450_25d	25	28	28	28	28	?	?
DSJC500.1	?	14	14	14	14	14	14
DSJC500.5	?	55	55	55	55	51	50

Table 4 presents some instances taken from the Carter's Benchmark [5]. We again present the number of vertices, edges and edge density of these graphs in this table. Table 5 represents the best colorings obtained after 50 runs. In Figure 4, the average colorings of 50 runs are presented.

For instances in the Carter's timetabling benchmark, again, a significant difference among crossover operators is not observed. However, LIMX has a slightly better performance in terms of best and average color (group) number. LIMX gave the best colorings in staf83 and lsef91 instances while others were one color behind it. Yet, the difference between average colorings and standard deviation is not statistically significant for almost all instances.

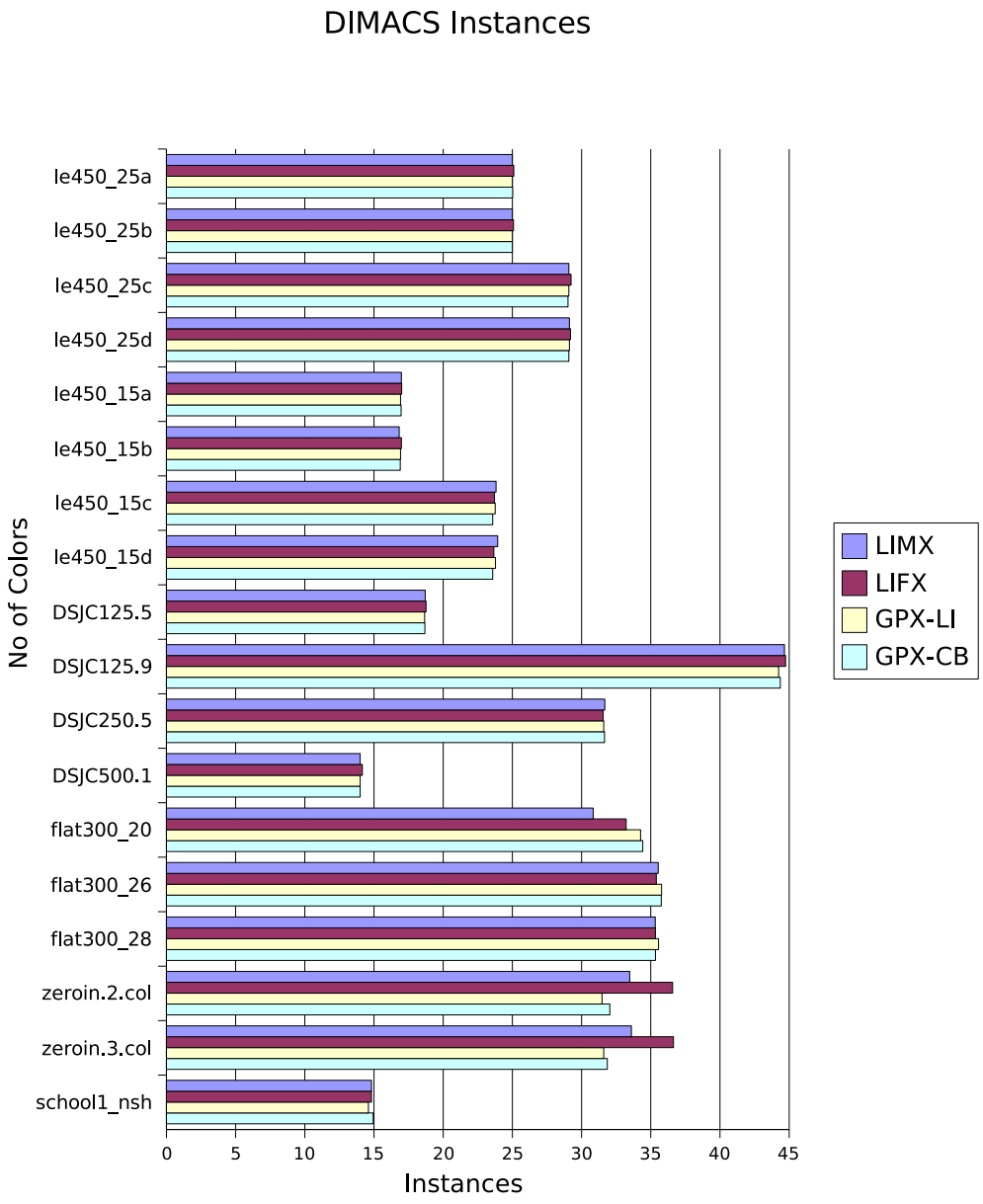

Fig. 3. Average number of groups for instances in DIMACS Benchmark.

Table 4. Data Characteristics of the problem instances from the Carter Benchmark Suite

Instance	$\|V\|$	$\|E\|$	%
Hecs92	81	1363	0.42
Staf83	139	1381	0.14
Yorf83	181	4691	0.29
Utes92	184	1430	0.08
Earf83	190	4793	0.27
Tres92	261	6131	0.18
Lsef91	381	4531	0.06
Kfus93	461	5893	0.06
Ryes93	486	8872	0.08
Carf92	543	20305	0.14
Utas92	622	24249	0.13
Cars91	682	29814	0.13

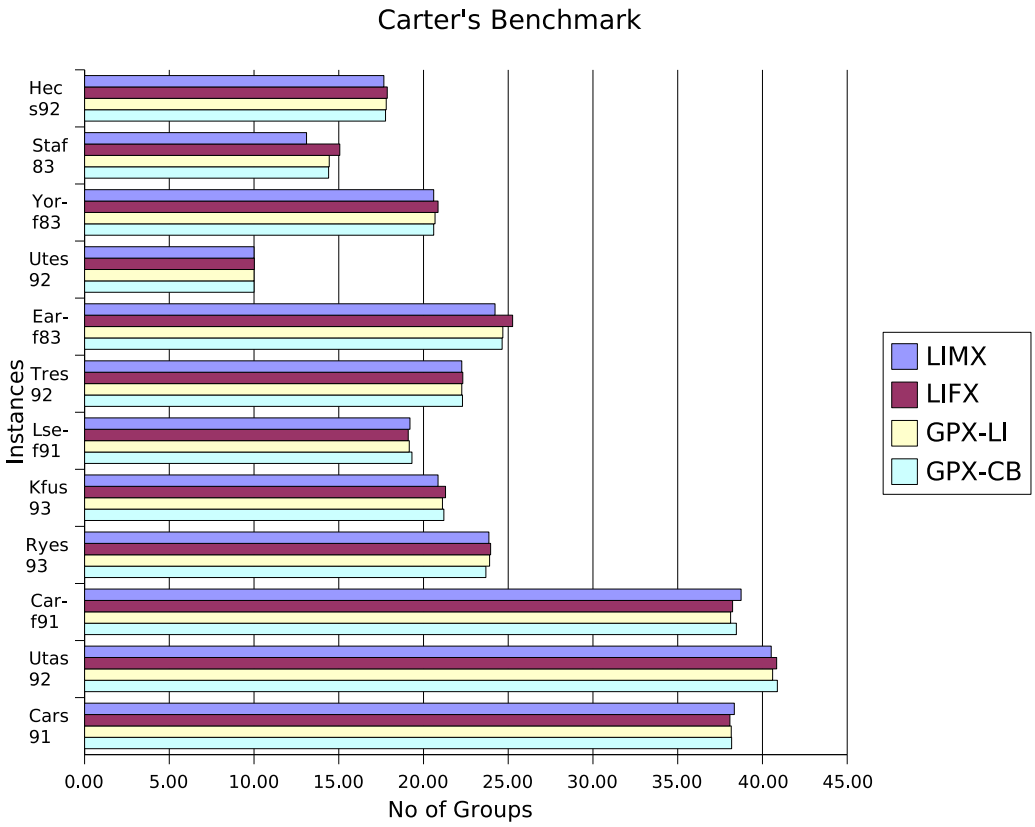

Fig. 4. Average number of Groups obtained for instances in Carter's Benchmark.

Table 5. Best colorings obtained for the instances in the Carter's Benchmark Suite

Instance	LIMX	LIFX	GPX-LI	GPX-CB	Carter	Caramia	Merlot
Hecs92	17	17	17	17	17	17	18
Staf83	13	14	14	14	13	13	13
Yorf83	20	20	20	20	19	19	23
Utes92	10	10	10	10	10	10	11
Earf83	23	24	24	23	22	22	24
Tres92	21	21	21	21	20	20	21
Lsef91	17	18	18	18	17	17	18
Kfus93	20	20	20	20	19	19	21
Ryes93	23	23	23	23	21	21	22
Carf92	36	36	36	36	28	28	31
Utas92	38	39	38	38	32	30	32
Cars91	36	36	37	35	28	28	30

We have also compared the best colorings after 10000 generations with some of the results from the literature (Carter et. al [5], Caramia et. al. [4] and Merlot et. al [21]). Like DIMACS instances, the performance of the graphs with vertices above 500 suffered due to the limit on the maximum number of generations. For instances, our crossovers gave similar results in terms of best grouping obtained. Generally they obtained colorings equal or one color behind colorings of Carter et. al and Caramia et. al, and better than of Merlot et. al.

Full test results are presented with average best coloring found, standard deviation, and best coloring of 50 runs for each crossover operator in Table 6.

5 Conclusion

In this paper, we have investigated the performance of LLE in well known grouping problems, exam timetabling and graph coloring. We presented several crossover operators that can be used with LLE. The results obtained are promising since LIMX and LIFX perform approximately similar to the two variants of GPX, which is an integral part of the most successful graph coloring algorithm. Also our crossover operators gave satisfactory results for instances in Carter's and DIMACS benchmark suites. In the future Linear Linkage Encoding will be used on other grouping problems together with the crossover operators aforementioned. The multi-objective LLE framework will be used for timetabling problems with additional constraints.

6 Acknowledgements

This research is funded by TUBITAK (The Scientific and Technological Research Council of Turkey) under grant number 105E027.

References

1. Avanthay C., Hertz A. and Zufferey N. (2003) Variable neighborhood search for graph coloring, European Journal of Operational Research 151 pp 379-388.
2. Brelaz, D. (1979) ”New Methods to Color Vertices of a Graph”, Communications of the ACM 22 pp 251-256.
3. Burke, E.K., Newall, J. and Weare, R.F. (1996) ”A Memetic Algorithm for University Exam Timetabling”. In: Burke, E.; Ross, P. (eds.): Practice and Theory of Automated Timetabling, First International Conference, Edinburgh, U.K., August/September 1995. Selected Papers. Lecture Notes in Computer Science 1153, Springer-Verlag, Berlin Heidelberg New York 241–250.
4. Caramia, M., Dell’Olmo, P., and Italiano, G.F. (2000) ”New algorithms for examination timetabling”. In: Nher, S.; Wagner, D., (eds.): Algorithm Engineering 4th International Workshop, WAE 2000, Saarbrucken, Germany, September 2000. Proceedings. Lecture Notes in Computer Science 1982, Springer-Verlag, Berlin Heidelberg New York, 2001, 230-241.
5. Carter, M. W, Laporte, G. and Lee, S.T. (1996) ”Examination timetabling: algorithmic strategies and applications.”, Journal of the Operational Research Society, 47:373-383.
6. Du, J., Korkmaz E., Alhajj R., Barker K., (2004) ”Novel Clustering Approach that Employs Genetic Algorithm with New Representation Scheme and Multiple Objectives.” In Proceedings of 6th International Conference on Data Warehousing and Knowledge Discovery - DaWaK ’04, Zaragoza, Spain.
7. Even, S., Itai, A. and Shamir, A. (1976) ”On the Complexity of Timetable and Multicommodity Flow Problems”, SIAM J. Comput., 5(4):691-703.
8. Falkenauer, E. (1998) ”Genetic Algorithms and Grouping Problems”, John Wiley & Sons.
9. Fleurent, C. and Ferland, J.A. (1996) ”Genetic and Hybrid Algorithms for Graph Coloring”, Annals of Operations Research 63 pp 437-461.
10. Galinier, P. and Hao, J.K., (1999) ”Hybrid Evolutionary Algorithms for Graph Coloring”, Journal of Combinatorial Optimization 3 pp 379-397.
11. Garey, M.R., Johnson, D.S., (1979) ”Computers and Intractability: A Guide to the Theory of NP-Completeness”, W.H. Freedman San Francisco.
12. Goldberg, D.E, (1989) ”Genetic Algorithms in Search, Optimization, and Machine Learning”, Addison Wesley, Reading, Massachussetts.
13. Holland, J. (1975) ”Adaptation in Natural and Artificial Systems”, University of Michigan Press
14. Hertz, A., and De Werra, D. (1987) ”Using Tabu Search Techniques for Graph Coloring”, Computing, 39 pp 345-351.
15. Horn, J., Nafpliotis, N., Goldberg, D. E., (1994), ”A Niched Pareto Genetic Algorithm for Multiobjective Optimization”, Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, vol 1. pp 82-87, Piscataway, New Jersey.
16. Johnson, D.S, Aragon, C.R, McGeoch, L.A, and Schevon, C., (1991) ”Optimization by Simulated Annealing: An Experimental Evaluation: Part II, Graph Coloring and Number Partitioning”, Operations Research, 39 (3) pp 378-406.
17. Kirovski D., and Potkonjak. M., (1998), Efficient Coloring of a Large Spectrum of Graphs. 35th Design Automation Conference Proceedings, pp. 427-432.
18. Johnson, D.S, Trick, M.A, (Editors), (1996) ”Cliques, Coloring and Satisfiability”, DIMACS Series in Discrete Mathematics and Theoretical Computer Science”, Vol. 26, American Mathematical Society.
19. Leighton, F.T, (1979) ”A Graph Coloring Algorithm for large Scheduling Problems”, Journal of Research of the National Bureau Standard, 84 pp 79-100.

20. Terashima-Marn, H., Ross, P. and Valenzuela-Rendn, M. (1999) ”Clique-Based Crossover for Solving the Timetabling Problem with Gas”, Proc. of the Congress on Evolutionary Computation, pp. 1200-1206.
21. Merlot, L.T.G., Boland, N., Hughes B.D., and Stuckey, P.J. (2002) ”A Hybrid Algorithm for the Examination Timetabling Problem.” In: Burke, E.; De Causmaecker, P. (eds.): Proceedings of Practice and Theory of Automated Timetabling, Fourth International Conference, Gent, Belgium, August 2002. pp 348-371.
22. Ozcan, E., Ersoy, E., (2005)”Final Exam Scheduler - FES”, Proc. of 2005 IEEE Congress on Evolutionary Computation, vol. 2, pp 1356-1363.
23. Paquete, L. F. and Fonseca C. M. (2001) ”A Study of Examination Timetabling with Multiobjective Evolutionary Algorithms”, Proc. of the 4th Metaheuristics International Conference (MIC 2001), pp. 149-154, Porto.
24. Radcliffe, N.J, ”Formal Analysis and random respectful recombination” (1991), In Proceedings of the 4th International Conference on Genetic Algorithm, pp. 222-229
25. Ramani, A., Aloul F.A, Markov, I and Sakallah, K.A,. ”Breaking Instance-Independent Symmetries in Exact Graph Coloring.” (2004) Design Automation and Test Conference in Europe, pp 324-329
26. Wong, T., Cote, P. and Gely, P. (2002) ”Final exam timetabling: a practical approach”, Canadian Conference on Electrical and Computer Engineering, Winnipeg, CA, vol.2, pp. 726- 731.

Table 6. Average Best, Standard Deviation and Best Results for DIMACS and Carter's Benchmark

		LIMX			LIFX			GPX-LI			GPX-CB		
Instance	$\chi(G)$	μ	σ	B	μ	σ	B	μ	σ	B	μ	σ	B
DSJC125.5	?	18,70	0,46	18	18,76	0,43	18	18,66	0,47	18	18,68	0,47	18
DSJC125.9	?	44,66	0,47	44	44,74	0,48	44	44,24	0,43	44	44,36	0,48	44
zeroin.3.col	30	33,60	1,33	31	36,64	1,02	35	31,60	0,94	30	31,86	0,90	31
zeroin.1.col	49	49,04	0,20	49	50,66	0,55	50	49,00	0,00	49	49,00	0,00	49
zeroin.2.col	30	33,48	1,41	31	36,58	1,02	35	31,48	0,92	31	32,04	1,22	31
DSJC250.1	?	9,00	0,00	9	9,00	0,00	9	9,00	0,00	9	9,00	0,00	9
DSJC250.5	?	31,68	0,58	31	31,56	0,54	31	31,60	0,57	31	31,66	0,47	31
DSJC250.9	?	75,38	0,56	75	75,90	0,70	75	75,38	0,77	75	75,42	0,70	74
flat300_20	20	30,84	3,85	20	33,22	1,30	31	34,26	1,55	27	34,42	1,33	32
flat300_26	26	35,54	0,57	34	35,40	0,53	34	35,78	0,64	34	35,76	0,64	34
flat300_28	28	35,32	0,51	34	35,34	0,51	34	35,56	0,64	34	35,34	0,65	34
school1_nsh	14	14,82	0,82	14	14,82	0,89	14	14,60	0,80	14	14,94	0,95	14
le450_15a	15	16,98	0,32	16	17,00	0,20	16	16,92	0,27	16	16,96	0,20	16
le450_15b	15	16,82	0,43	16	16,98	0,14	16	16,92	0,27	16	16,90	0,30	16
le450_15c	15	23,82	0,52	23	23,70	0,46	23	23,76	0,43	23	23,58	0,49	23
le450_15d	15	23,94	0,47	23	23,66	0,51	23	23,78	0,50	23	23,58	0,53	23
le450_25a	25	25,00	0,00	25	25,10	0,30	25	25,00	0,00	25	25,02	0,00	25
le450_25b	25	25,00	0,00	25	25,08	0,27	25	25,00	0,00	25	25,00	0,00	25
le450_25c	25	29,08	0,34	28	29,24	0,43	29	29,08	0,34	28	29,02	0,32	28
le450_25d	25	29,12	0,43	28	29,20	0,45	28	29,12	0,47	28	29,08	0,34	28
DSJC500.1	?	14,00	0,00	14	14,16	0,37	14	14,00	0,00	14	14,00	0,00	14
DSJC500.5	?	55,98	0,55	55	55,96	0,49	55	56,00	0,57	55	56,18	0,71	55
Hecs92	?	17,66	0,47	17	17,86	0,4	17	17,80	0,40	17	17,76	0,43	17
Staf83	?	13,10	0,30	13	15,06	0,37	14	14,44	0,57	14	14,40	0,49	14
Yorf83	?	20,60	0,53	20	20,86	0,45	20	20,68	0,51	20	20,60	0,60	20
Utes92	?	10,00	0,00	10	10,02	0,14	10	10,00	0,00	10	10,00	0,00	10
Earf83	?	24,22	0,76	23	25,26	0,72	24	24,68	0,55	24	24,64	0,82	23
Tres92	?	22,26	0,52	21	22,32	0,51	21	22,26	0,56	21	22,30	0,57	21
Lsef91	?	19,20	0,66	17	19,10	0,67	18	19,16	0,67	18	19,32	0,58	18
Kfus93	?	20,86	0,57	20	21,30	0,61	20	21,12	0,65	20	21,20	0,57	20
Ryes93	?	23,86	0,49	23	23,96	0,66	23	23,90	0,64	23	23,68	0,55	23
Carf92	?	38,74	1,29	36	38,24	1,01	36	38,12	1,05	36	38,46	1,05	36
Utas92	?	40,52	1,03	38	40,84	1,01	39	40,60	1,08	38	40,88	1,01	38
Cars91	?	38,34	1,16	36	38,08	1,00	36	38,16	1,03	37	38,18	1,23	35