Available online at www.sciencedirect.com
ScienceDirect
European Journal of Operational Research 179 (2007) 759–774
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/ejor

A simulated annealing based hyperheuristic for determining shipper sizes for storage and transportation

Kathryn A. Dowsland [a,b,*], Eric Soubeiga [a], Edmund Burke [a]

[a] The University of Nottingham, The School of Computer Science and IT, Wollaton Road, Nottingham NG8 1BB, United Kingdom
[b] Gower Optimal Algorithms Ltd., 5 Whitestone Lane, Newton, Swansea SA3 4UH, United Kingdom

Received 5 April 2004; accepted 23 March 2005
Available online 29 November 2005

Abstract

The current drive to reduce packaging waste has led many companies to consider the use of multi-trip containers or shippers in which to transport their products in order to reduce packaging waste. The efficiency of such systems obviously depends on selecting shipper dimensions in such a way as to ensure high volumetric utilisation. As is the case with many practical problems the efficiency/solution quality can be improved if problem specific information is used to enhance the operation of a meta-heuristic solution approach. The problem can be modelled as a *p*-median problem but is too large to be solved in reasonable time without further modification. Four such modifications, all based on properties of the physical problem, are introduced and incorporated into a hyperheuristic driven simulated annealing solution approach.

Keywords: Packing; Logistics; Simulated annealing; Hyperheuristics

1. Introduction

The ongoing drive to reduce packaging waste has led to an increasing interest in the use of multi-trip containers or shippers. The usual practice with disposable containers is to use a bespoke carton for each product. However, in a multi-use system it is often necessary to reduce this to a relatively small number in order to maintain a manageable and cost effective logistics system for the containers themselves. This will inevitably result in wasted space within the containers, potentially increasing both the monetary and

* Corresponding author. Address: Gower Optimal Algorithms Ltd., 5 Whitestone Lane, Newton, Swansea SA3 4UH, United Kingdom. Tel.: +44 1792 368413; fax: +44 0115 8467591.
E-mail addresses: kad@cs.nott.ac.uk (K.A. Dowsland), exs@cs.nott.ac.uk (E. Soubeiga), ekb@cs.nott.ac.uk (E. Burke).

0377-2217/$ - see front matter
doi:10.1016/j.ejor.2005.03.058

environmental costs. Thus it is important that the container dimensions be chosen in such a way as to minimise such wastage. This paper deals with the design and evaluation of a heuristic solution to the problem of selecting a set of shippers that minimises the total annual volume of space required to accommodate a given set of products with known annual shipment quantities.

The precise definition of the problem will vary from company to company as it will depend on a number of factors. These include features of the individual products (e.g. product density, whether or not it needs to be kept 'right way up' etc.), handling restrictions (e.g. weight restrictions, restrictions on dimensions imposed by conveyers etc.) and other features of the distribution system (e.g. if palletisation utilisation or loading efficiency in trucks or shipping containers is to be taken into account). Our aim is therefore to develop an approach that is sufficiently flexible to deal with such variations.

In the next section we show that the problem has a natural model as a p-median problem, but that the typical problem size and the computational expense of calculating the cost function coefficients means that it is not feasible to apply existing exact or heuristic p-median algorithms directly to the problem. We then go on to show how these difficulties can be overcome within the context of a local search approach to the problem. This is achieved by using problem specific information to make four improvements: the introduction of a pre-processing phase to speed up cost function calculation, the reduction of the size of the solution space, the partition of the natural neighbourhood into a set of more manageable sub-neighbourhoods, and the addition of a fast post-processing phase. The first three enhancements result in a considerable reduction in the time taken for the convergence of a descent based local search approach to the problem. As a result of this improvement we were able to use the underlying local search framework as the basis of a simulated annealing type algorithm. However questions are raised as to which of several possible neighbourhood structures should be adopted and how the neighbourhood should be sampled. Answering these questions empirically by running a sufficient number of trials for each possibility would require a vast amount of computational effort and may not result in a definitive solution, as the best choice may depend on the characteristics of the data or the precise objectives and constraints of the problem. Therefore we adopt a different strategy and use the concept of a hyperheuristic (Burke et al., 2003b) to control the neighbourhoods and sampling rules at each stage of the search. The original tabu-search hyperheuristic as suggested in Burke et al. (2003a) was not designed for a simulated annealing environment and cannot be applied directly to an implementation such as ours. We outline the reasons for this and suggest suitable modifications. The most promising of these is incorporated into our final implementation. Real data from a cosmetics company is used to evaluate the different approaches and to show that the final implementation is consistently able to provide good solutions with a very reasonable amount of computational effort.

It is also worth noting that there has been a significant level of recent research effort in exploring effective search methodologies for a wide range of other cutting and packing problems. Recent papers have explored issues in one dimensional packing (Leung et al., 2001), two dimensional problems (Lodi et al., 2003; Hopper and Turton, 2001; Burke et al., 2004; Mukhacheva and Mukhacheva, 2004) and in three dimensions (Cagan et al., 2002). Other papers have concentrated on the solution of real-life problems focussing on high quality solutions across a variety of industries and application areas which include leather cutting (Crispin et al., 2003), the clothing industry (Yeung and Tang, 2003), the paper industry (Menon and Schrage, 2002), the furniture industry (Morabito and Arenales, 2000) and glass cutting (Puchinger et al., 2004).

2. The problem

Our problem can be stated as follows. A company produces a set of n different products. Associated with each product, $i(i = 1, n)$, are the four physical characteristics of length, width, height and weight, denoted x_i, y_i, z_i and w_i, respectively. Each product also has an expected annual sales volume in terms of number of units sold, denoted by b_i. The objective, for a given integer $p(p < n)$, is to find a set of p totes or shippers

$j(j = 1, p)$ to be used for storage and/or distribution of the products, such that the annual utilisation of shipper space is maximised. The dimensions of shipper j will be represented by variables X_j, Y_j and Z_j. As the expected total volume of products shipped is fixed, the problem is equivalent to that of minimising the total volume of shippers required. It is assumed that each individual shipper is filled with a single product (although shippers of a given set of dimensions may obviously be used for several different products). There is a weight limit on the amount of product packed into each shipper. For the purposes of this study we also assume that the product is packed into the shipper in identical layers with the height dimension placed vertically. The loading patterns for the individual layers are those obtained using the PALLETMANAGER pallet loading software {www.packyourpallet.com}, which uses an extension of the heuristic described in Bischoff and Dowsland (1982). The extensions, based on Dowsland (1987a), have been shown to guarantee optimal solutions for all problems satisfying certain limits on the dimensions and aspect ratios of the boxes and container. It is worth noting that all problems solved during the experiments quoted in this paper fall within this range. Thus all the pallet loading solutions are optimal and could have been obtained (albeit with an increase in computational time) using the algorithm in Dowsland (1987a). However, as discussed later, the approach can be easily adapted to less stringent assumptions.

The problem has a natural model as a p-median problem in which the supply locations are given by the set of m feasible shipper dimensions and the demand locations by the set of n products. The cost of allocating product i to shipper j is the product of the volume of shipper j and the number of shippers that would be required to accommodate the annual demand for product i. The problem can then be stated as:

$$\min \quad \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij}$$

$$\text{s.t.} \quad \sum_{j=1}^{m} x_{ij} = 1, \quad i = 1, n$$

$$x_{ij} \leqslant y_j, \quad i = 1, n, \ \ j = 1, m$$

$$\sum_{j=1}^{m} y_j = p$$

$$x_{ij}, y_i \in (0, 1),$$

where $x_{ij} = 1$ if product i is allocated to shipper j, $x_{ij} = 0$ otherwise, $y_j = 1$ if shipper j is used, $y_j = 0$ otherwise, and $c_{ij} = \frac{V_j b_i}{f_{ij}}$ where $V_j = X_j * Y_j * Z_j$ and f_{ij} is the number of units of product i that will fit into shipper j. If we denote the maximum weight per shipper by $W_{\max}$ and define t_{ij} to be the number of copies of a box of base dimensions x_i by y_i that will fit into a single layer on a loading area of dimensions X_j by Y_j then $f_{ij} = \min\left\{\left\lfloor\frac{W_{\max}}{w_i}\right\rfloor, \left\lfloor\frac{Z_j}{z_i}\right\rfloor * t_{ij}\right\}$.

The p-median problem has been widely studied and a variety of different approaches have been suggested for its solution. These include both exact and heuristic approaches. Exact approaches are usually based on the integer programming formulation of the problem such as the Lagrangian relaxation algorithms outlined in Beasley (1985) or the dual approach of Hanjoul and Peeters (1985), and are capable of routinely solving moderately sized problems with $n \leqslant 500$ and $p \leqslant 50$. Many of the more successful heuristic approaches are neighbourhood search type algorithms based on the original 1-opt descent method of Teitz and Bart (1968). Variations include using add and drop neighbourhoods instead of the original swap neighbourhood (Whitaker, 1983) and widening the scope of the search using more modern approaches such as variable neighbourhood search (Hansen and Mladenovic, 1997), tabu search and strategic oscillation (Rolland et al., 1996) and GRASP (Resende and Werneck, 2004). Typically these report good results for problems with values of n in the low thousands. Implementations based on other heuristics such as GAs (Estivill-Castro

and Torres-Velazquez, 1999), dynamic programming (Hribar and Daskin, 1997) and scatter search (Garcia-Lopez et al., 2003) have also been suggested.

However, in our case the size and complexity of the problem precludes the direct application of these approaches. Although the value of n, the number of products, and the value of p, the number of shippers, are both likely to be relatively small when compared with the maximum size of problems quoted in many of the above papers, the value of m, the potential number of shipper sizes (usually assumed to be no bigger than n), is extremely large. Test data dimensions from the company were supplied to the accuracy of one-tenth of a millimetre. Even if we assume that millimetre accuracy is sufficient, allowing shipper dimensions to range from 100 mm to 600 mm results in a value of m of the order of 10^8. Thus any solution approach that requires all constants c_{ij} to be evaluated is not feasible. This is not a problem for approaches based on local search in which only the costs of those solutions visited need to be evaluated. The natural local search framework for the problem is that introduced by Teitz and Bart and used as the basis of most of the published neighbourhood approaches. The solution space is defined as the set of subsets of exactly p supply locations (i.e. shippers) and a neighbourhood move is defined by swapping a location in the current subset for one that is not in the subset. However, it was felt that any approach based directly on this framework without further enhancement was unlikely to be successful due to the combined effect of the following factors. Firstly the rationale behind traditional local search is to use the neighbourhood structure to ensure that only a small part of search space needs to be considered at one time. More recently it has been shown that it may be advantageous to use very large neighbourhoods (Orlin et al., 2002), but only if good solutions within the neighbourhood can be located efficiently, for example by using dynamic programming (Congram et al., 2002) or some other optimisation procedure appropriate to the neighbourhood structure. In our case the size of m relative to a small value of p will mean that each neighbourhood covers a large part of the solution space. Indeed the neighbourhoods are so large that an exhaustive search of a single neighbourhood will be computationally infeasible, and there is no obvious optimisation technique that will improve matters. Secondly the overall size of the solution space is given by $\binom{m}{p}$. When $m = 10^8$ and $p = 10$ this will be of the order of 10^{64} and a large number of iterations are likely to be needed to search the space adequately. Thirdly the calculation of each c_{ij} requires the solution of the pallet-loading problem (i.e. that of maximising the number of copies of product i that can be fitted on a single layer of a shipper j), which is in itself a time consuming operation. Each trial move will require the solution of n such problems.

Therefore, the performance of any local search approach to the problem is likely to be enhanced by any of the following:

1. A neighbourhood structure consisting of smaller neighbourhoods.
2. A reduction in the size of the solution space.
3. A means of speeding up the calculation of c_{ij} (or more specifically the calculation of t_{ij}).

In the next section we consider each of these in turn and show how all three can be achieved by utilising information from the original problem.

3. Modifying a local search implementation with problem specific information

The basis of our local search framework is a random descent strategy (i.e. accepting an improving move as soon as it is sampled) using the solution space, neighbourhood structure and evaluation function used by Teitz and Bart. Thus the solution space consists of all subsets of p shippers, a neighbourhood move involves changing the size of a single shipper, and a move is accepted if this results in a reduction in cost. Because of

the large size of the neighbourhoods involved the search is run for a fixed number of iterations rather than continuing until the whole neighbourhood has been sampled without any improvement. As outlined above this will result in a very large search space with neighbourhoods of size $(m - p) \cdot p$ (i.e. of the order of 10^8) requiring the solution of n pallet loading problems at each iteration. Due to the stopping condition the solution may not correspond to a local optimum and it is possible that in some cases it may be obvious that a particular shipper does not provide a good fit for any of the products allocated to it. In the following subsections we address each of these issues in turn and show how information from the original problem can be used to make a series of improvements.

3.1. Reducing the computational burden of cost calculation

Our strategy for reducing the cost function calculation is based on the fact that the set of instances of the pallet-loading problem can be partitioned into a set of equivalence classes, such that all the instances in each class have the same set of optimal solutions. Thus all that is required is to determine the finite set of equivalence classes that cover the given set of products and feasible range of shipper dimensions, and to solve one member of each. As long as we have an appropriate means of indexing the classes so that an arbitrary instance can be assigned to the appropriate class we can then use a look-up table to find t_{ij} and hence simplify the calculation of c_{ij}. This is achieved as follows.

In Dowsland (1985) and Dowsland (1987a) it is shown that the equivalence classes are defined by the combinations of product lengths and widths—say x and y—that will fit into the dimensions of the loading area. More formally, let $f(k_1, k_2) = k_1x + k_2y$ for any non-negative integers k_1, k_2. Given three constants x, y and S, with $y \leqslant x \ll S$, we define the set of efficient partitions of S in x and y by $E(S, x, y) = \{d = f(k_1, k_2)\colon d \leqslant S < f(k_1, k_2 + 1)\}$. It is shown that two pallet loading problems will have the same set of feasible layouts if they have the same set of efficient partitions of the box dimensions in both of the loading dimensions. In Dowsland (1987b) a method is presented for indexing the equivalence classes in the general case. For our purposes this can be simplified as, for each product i, we are only interested in those members of the equivalence class with constant x_i and y_i. Consider a point S' such that $S' = f(k_1, k_2)$. Now suppose we gradually increase S' by a small amount ε. The efficient partitions of $S' + \varepsilon$ in x_i and y_i will remain the same as those for S' until ε reaches the point where $S' + \varepsilon = f(k_3, k_4)$ for some k_3 and k_4, when a new efficient partition becomes valid. We can therefore define the unique members of each equivalence class as those rectangles X by Y where both X and Y are linear combinations of the product dimensions. These can be indexed by the quadruple $(k_{1X}, k_{2X}, k_{1Y}, k_{2Y})$ where $X = f(k_{1X}, k_{2X})$ and $Y = f(k_{1Y}, k_{2Y})$. In order to look-up the correct value of t_{ij} all that is required is to find coefficients satisfying $f(k_{1X}, k_{2X}) = \max\{f(k_1, k_2)\colon f(k_1, k_2) \leqslant X_j\}$, together with the corresponding coefficients for Y_j. This is easily achieved by letting k_1 range from 0 to $\text{int}(X_j/x_i)$, setting $k_2 = \text{int}((X_j - k_1x_i)/y_i)$ and selecting the pair that maximises f.

3.2. Reducing the neighbourhoods

The physical problem also suggests two natural ways of reducing the size of the neighbourhoods. The Teitz and Bart neighbourhood corresponds to changing the dimensions of one of the shippers. This implies changing all three dimensions at once. An obvious way to define smaller neighbourhoods is to change just one or two dimensions keeping the remaining side(s) unchanged. This also makes sense in terms of the search as a moderately good solution may be achieved if the products fit snugly into one or more dimensions but less well in the others. A better fit will be achieved if the remaining dimension(s) are changed to a more appropriate value. With the original neighbourhood the majority of neighbours would destroy these 'good' dimensions whereas with a neighbourhood based on changing fewer dimensions there is a greater chance that these would be preserved. Assuming millimetre accuracy and shipper dimensions in the range

100–600 mm and $p = 10$ the original neighbourhood would be of size 1.25×10^9 while a neighbourhood based on changing one dimension would only be of size 1.5×10^4. As the neighbourhood moves involve changing one or more dimensions a second way of reducing their size is to limit the magnitude of such changes to be less than some constant d. Once again this makes sense, especially if d is small. If we increase or decrease one of the shipper dimensions gradually then the c_{ij} values will also change by a small amount until either a new efficient partition is achieved or an existing partition is lost for some product i. In the latter case, this may result in a sharp increase in solution cost and so it is likely that this reduction will only be successful if the search mechanism incorporates some means for accepting uphill moves as in simulated annealing or tabu search.

3.3. Reducing the size of the solution space

As well as providing a means of speeding up the cost function calculation the concept of efficient partitions can be used to reduce the size of the solution space. It is clear that there is no point in including shipper dimensions such that every product would leave a gap if packed into a shipper of that size. In the case of the height dimension this means that we need only consider heights of the form $Z = k \cdot z_i$ for some positive integer k and some product i. In the case of the base it is shown in Dowsland (1985) that for a product of dimensions x_i by y_i if X, X' and Y, Y' satisfy $X' = \max\{f(k_1, k_2): f(k_1, k_2) \leqslant X\}$ and $Y' = \max\{f(k_1, k_2): f(k_1, k_2) \leqslant Y\}$ then any layout of i on a loading area of dimensions X by Y will also fit into a loading area X' by Y'. Thus the only base shipper dimensions that need to be considered are those that form efficient partitions with respect to the dimensions of some product. The amount of reduction this gives will depend on the precise product dimensions but for the data used in the experiments described in the next section m was reduced from 1.25×10^{11} to 1.25×10^8, implying a reduction factor of the order of 10^{3p} in the solution space for a problem with p shippers.

3.4. A fast post-processor

The above argument relates to the set of dimensions that are worth considering. However if we have a given solution then each product will be allocated to a particular shipper, and we can take this reduction process one stage further, as there is no point in using a shipper with any dimension that is not an efficient partition of the products allocated to it. We can check this for each solution and reduce the dimensions accordingly. However it may not be advantageous to apply this after every move for two reasons. Firstly if one or more dimensions have been reduced it is possible that the current allocation of products is no longer optimal and so some products may need to be re-allocated. This in turn could lead to another round of reductions for those shippers from which products have been moved. Thus the process is not trivial in terms of computational cost. Secondly it is possible that such a reduction may preclude future moves that would have been accepted without the reduction, thus making the search less flexible and more prone to convergence in poor local optima. Experiments were carried out to evaluate the effects of applying this type of reduction after every move, after a fixed number of moves, or at the end. All results suggested that there was no benefit in reducing dimensions in this way during the search, but that it could be beneficial as a post-processing phase to improve the final solution. Therefore throughout the remainder of this paper we assume that the final solution has been reduced in this way.

4. Initial experiments

Our initial experiments were designed to measure the effects of the strategies outlined in the previous section and to provide the basis for deciding on an appropriate search strategy for our final implementation.

They all use straightforward random descent in which an initial set of p shippers is obtained by sampling the three dimensions independently from the appropriate range and potential moves are sampled randomly from the neighbourhood of the current solution. An improving move is accepted as soon as it is found. A data set of 46 products was provided by a cosmetics company with all three dimensions measured in mm to one decimal place. In all experiments we assume that the shipper dimensions range from 100 mm to 600 mm also to one decimal place, and that there is a weight limit per shipper of 20 kg, as given by the company. All experiments in this section use $p = 3$, 6 and 9.

Three neighbourhood structures were defined as follows. N3 is the original neighbourhood in which all three dimensions are changed, N2 involves changing either both base dimensions or the height dimension, and N1 involves changing just one dimension. In the case of N1 and N2 one of the three possible dimensions is selected with equal probability. For N2, if this is one of the base dimensions then the other base dimension is also changed. Thus for N2 the probability of changing the base is twice that of changing just the height.

Experiment 1 was designed to measure the effect of the pre-processor and involved five random starts using the N3 neighbourhood for each of the three values of p run for 200 000 iterations, with and without the cost function pre-processor. The total time for the run without the pre-processor was 7221 seconds, while that with the pre-processor was only 494 seconds. With the pre-processor the total number of pallet loading problems that needed to be evaluated per run was 41 916, ranging from just 45 for the largest product to 7287 for the smallest, as compared with the 9 200 000 problems that are required for 200 000 iterations and 46 products. Such savings are clearly worthwhile and will increase as the number of iterations increases. All future experiments are carried out using the pre-processor.

The main set of experiments in this section involves evaluating the three different neighbourhoods. For each value of p the search was run for 20 random starts over 2 000 000 iterations with the results being recorded after 200 000 iterations and 2 000 000 iterations. These are given in Table 1 with the best result for each value of p marked in bold. Note that while the search uses the annual shipper volume as its cost function, the results are expressed in terms of percentage volumetric utilisation of the shippers, as this is the measure usually required in practice.

A number of trends are apparent from the results. Firstly the best results both in terms of the mean and the best result out of the 20 runs always occur using neighbourhood N2. Secondly there is clear evidence that 200 000 iterations are sufficient for the runs using N1 to converge. However the results are not as good as those

Table 1
Space utilisation obtained with random descent for $p = 3$, 6, 9 for different neighbourhoods and solution space definitions

Neighbourhood	Iterations	Solution space	$p = 3$			$p = 6$			$p = 9$		
			Mean	Max	Last move[a] ('000s)	Mean	Max	Last move[a] ('000s)	Mean	Max	Last move[a] ('000s)
N1	2×10^5	Full	88.03	89.06		93.65	94.52		96.28	96.91	
		Reduced	87.77	89.06		93.65	94.27		96.14	96.91	
	2×10^6	Full	88.03	89.06	104	93.65	94.52	308	96.29	96.91	479
		Reduced	87.77	89.06	36	93.65	94.27	114	96.14	96.91	220
N2	2×10^5	Full	88.14	89.33		93.86	94.34		96.31	96.94	
		Reduced	87.89	88.90		93.78	94.28		96.28	96.77	
	2×10^6	Full	**88.44**	89.35	1601	**94.15**	94.59	1701	96.57	**97.05**	1893
		Reduced	88.14	**89.36**	1364	94.02	**94.63**	1614	**96.62**	97.02	1703
N3	2×10^5	Full	87.04	88.72		92.79	93.83		95.51	96.36	
		Reduced	87.16	88.75		92.92	93.81		95.36	96.16	
	2×10^6	Full	87.98	88.78	1402	93.71	94.35	1580	96.28	96.65	1580
		Reduced	88.10	88.79	1360	93.85	94.48	1693	96.26	96.51	1760

[a] Iteration number of last accepted move.

using N2. The results using N3 tend to be worse, probably because the N3 neighbourhood is not close to convergence in the number of iterations allowed. In terms of solution quality there is little to choose between the full and reduced solution space, with both giving identical results with the N1 neighbourhood. However when we look at the number of iterations to the last improvement it is clear that the reduced solution space results in faster convergence for both the N1 and N2 neighbourhoods. (The lack of convergence for the N3 neighbourhood means that this statistic is irrelevant for this case). The standard deviations for solution quality (not given in the table) vary from 0.15 to 0.78 with most figures over 0.3, suggesting that none of the approaches is really successful in homing in on near optimal solutions. Thus a search strategy that allows uphill moves may be more successful, and the results for the N1 neighbourhood suggests that these neighbourhoods may be sufficiently small to consider a more powerful approach such as tabu search or simulated annealing, particularly if used in conjunction with the reduced solution space. Of these, simulated annealing is the natural choice as the large size of the neighbourhoods is better suited to an approach based on sampling individual neighbours, rather than considering all (or a large proportion of) the neighbourhood at each iteration.

In order to implement any simulated annealing algorithm for the shipper rationalisation problem a number of decisions need to be made. In our case the reduced solution space adds a further complication in that there is more than one natural way of sampling from this neighbourhood. Previous work using simulated annealing on other problems (Dowsland, 1993a,b; Thompson and Dowsland, 1998) suggests that sampling bias can have a significant effect on both solution time and quality. The results for the reduced space in Table 1 were obtained by sampling uniformly within the range 100–600 mm and then reducing the dimension to the nearest efficient partition dimension. As such dimensions are distributed more sparsely at the lower end of the scale this will mean that each efficient partition point relating to a smaller dimension will be sampled more frequently than each point relating to a larger dimension. This can obviously be overcome by sampling uniformly from the list of efficient partition dimensions. However this will mean that large container sizes will be sampled more frequently than small ones, which may or may not be desirable. Both strategies also have the potential disadvantage that points relating to efficient partitions with respect to more than one product will be sampled with the same frequency as those relating to just one product. Thus it may be advantageous to sample the products uniformly and then to sample an efficient partition relative to the product, or even to bias the sampling towards products that have larger shipment quantities. Once again this may introduce an undesirable bias in that large products have less efficient partitions than smaller ones and hence each partition for a larger product will be sampled more frequently than each partition for a smaller product.

Secondly, in spite of the observation concerning the convergence speed using N1 this may not be the best neighbourhood. Observations of the behaviour of the search during the descent process showed that moves relating to small changes in dimension were often made, suggesting that such a neighbourhood may be worth trying within the simulated annealing context. The overall performance of the N2 neighbourhood also suggests that this neighbourhood may be worth pursuing further. In order to find the best combination of sampling policy and neighbourhood structure a large amount of experimentation would be necessary. Even if such experimentation were to be carried out then it is not certain that the results would carry over to other variants of the problem and it is also possible that the best strategy may be some combination of neighbourhoods or sampling rules. We therefore chose to implement our SA algorithm in the form of a hyperheuristic in which neighbourhoods and sampling policies are selected dynamically during the search. However the tabu search hyperheuristic needs some adaptation in order to be used within a simulated annealing framework. This will be addressed in the next section.

5. The hyperheuristic in the context of SA

In essence a hyperheuristic is a high-level search strategy that selects from among a number of low-level heuristics at each point in the search. It can be thought of as a ‘‘heuristic to choose heuristics’’ (Burke et al.,

2003b) and as such a hyperheuristic approach operates on a search space of heuristics rather than the search space of problem solutions. It was motivated by the goal of finding a generic approach that could be applied to a variety of problems without the need for extensive problem specific development and testing. Further information on hyperheuristics and their application to a number of different problem areas can be found in Burke et al. (2003a), Burke et al. (2003b), Cowling et al. (2000), Cowling et al. (2002), Han and Kendall (2003), Kendall and Mohamad (2004), Kendall and Mohd Hussain (2004), Ross et al. (2003), Ross et al. (2002) and Soubeiga (2003). In the tabu-search hyperheuristics presented above (Burke et al., 2003a), each low-level heuristic is given a score based on an estimate of its performance potential, derived from information gathered during the search. In our Simulated Annealing, implementation the low-level heuristics are defined by different combinations of the 6 neighbourhoods and 5 different definitions of solution-space/sampling policy given below.

5.1. Neighbourhoods

N1, N2 and N3 as defined above, together with N1(10), N2(10), N3(10)—variations of N1, N2 and N3 in which the maximum allowed change in any dimension is ± 10 mm.

5.2. Sampling policies/solution space

S1: Uniform Euclidean sampling from the full solution space.
S2: Uniform Euclidean sampling from the full solution space followed by reduction to the nearest efficient partition.
S3: Uniform sampling from list of efficient partition points.
S4: Uniform sampling from the set of products, then uniform sampling of an efficient partition for that product.
S5: Roulette-wheel sampling from the set of products with probabilities proportional to annual shipment quantity, then uniform sampling of an efficient partition for that product.

These were combined to give the following 21 low-level heuristics: N1, N2, N3 with each of the 5 sampling policies, and N1(10), N2(10) and N3(10) with S1 and S2. Note that the restricted dimension neighbourhoods would be computationally complex to implement with the remaining sampling policies. In its simplest form the hyperheuristic uses information based on the performance of the low level heuristic used in the last application to calculate a score, $rk(h)$ for each low level heuristic h. If h resulted in an improvement its score is increased and if it resulted in a worsening of the objective its score is decreased. The basis of our implementation is that given in Burke et al. (2003a) and works as follows.

Algorithm HHTabu

Initialise:
Set tabu_status $h =$ '*false*' for each low-level heuristic, h.
Set max_tabu, the length of the tabu list, to an appropriate value.
Search:
Let h^* be the highest rank non-tabu low-level heuristic.
Apply h^* once. Let δ be the resulting improvement in the cost function.
Select low-level heuristic:
If $\delta > 0$ then
  $rk(h^*) = rk(h^*) + 1$
else

$rk(h^*) = rk(h^*) - 1$
set tabu_status(h^*) = '*true*'
if $\delta < 0$
set tabu_status(h) = '*false*' $\forall h \neq h^*$
else
if $|\{h$: tabu_status(h) = '*true*'$\}| >$ max_tabu let h' denote the heuristic that has been tabu the longest and set tabu_status(h') = '*false*'
endif
endif
Repeat search until stopping condition is reached.

HHTabu will continue to apply the selected low-level heuristic for as long as it continues to improve the solution, as its rank will be increased after each improving iteration. When the current choice fails to improve, its score is decreased and it is made tabu, so that a different heuristic will be chosen for the next iteration. If the solution is worse then all other low-level heuristics are released from the tabu list allowing the one with the highest score to be chosen. In order to avoid the process stagnating, if there is no change in the cost the tabu list will continue to be extended (up to a given limit), thus forcing some diversification in the choice of low-level heuristic.

However, it is not sensible to apply HHTabu directly within a simulated annealing environment for the following reasons. Firstly the proportion of improving moves is expected to be low, especially in the middle and later parts of the search. Thus there will be little opportunity for any of the low-level heuristics to improve their scores and all are likely to reduce quickly to zero. The obvious way of overcoming this is to define a single application of low-level heuristic h, to be k iterations of h. This raises the question as to what measures should be used to estimate δ as this can no longer be measured as the difference between the cost at the current solution and one of its neighbours. There are two natural ways to measure performance over k iterations. These are the best cost or the mean cost. The mean cost could be defined as the mean of the incumbent solutions over all iterations, the mean of the sampled solutions over all iterations, or the mean of the solutions over all iterations resulting in acceptance. Initial experiments based on the algorithm SAHHTabu as defined below, over 20 000 000 iterations and five starts for $p = 3$, 6 and 9, suggested that measuring the mean based on the sampled solutions resulted in bias towards the smaller neighbourhoods where the probability of sampling very bad solutions was lower, and that this was detrimental to solution quality. A similar, but far less pronounced trend was observed using the mean based on the incumbent or accepted solutions. Solution quality was slightly better when basing δ on the best solution encountered. As the objective of this research was to find an appropriate solution approach for the box rationalisation problem, and not to execute an in depth study of the use of HHTabu within the context of simulated annealing, this was adopted as the measure of δ for the remainder of the study without further experimentation.

The above solution will ensure that the low-level heuristics have a chance to improve their scores, but a second problem remains. A requirement of simulated annealing is that the volatility of the search, as measured in terms of changes in the cost function, decreases smoothly along with the temperature. At a particular temperature this volatility will certainly depend on the neighbourhood definition and may also be influenced by the sampling bias. This statement is supported by empirical evidence that showed that larger neighbourhoods require far higher temperatures to achieve a given rate of acceptance of uphill moves than their smaller counterparts. This suggests that, if a standard cooling schedule in which the temperature is reduced steadily is employed, then the relative performances of the low-level heuristics will depend more on the temperature than the state of the search in terms of the current position in the solution landscape. Thus an alternative strategy is needed.

In Dowsland (1993a,b) an undulating cooling schedule was introduced with the aim of producing a fairly constant acceptance rate in order to improve search performance for a class of problems with spiky solution

landscapes. The temperature was reduced after an accepted move according to the standard Lundy and Mees (1986) schedule using $t \to t/(1+\beta t)$ and reheated after a rejected move using $t \to t/(1-\alpha t)$. The required acceptance rate is approximated by $r = \alpha/\beta$. In Thompson and Dowsland (1996) it was shown that if r was reduced periodically then the same strategy could provide a flexible alternative to a standard cooling schedule, whilst adhering to the principle of reducing the volatility of the system over the length of the search. In the context of our hyperheuristic approach this means of adjusting temperature, as opposed to reducing it directly, should allow the volatility of the system to reduce smoothly regardless of the neighbourhood, as long as each low level heuristic application contains enough iterations for the acceptance rate to stabilise. These ideas have been incorporated into our simulated annealing hyperheuristic SAHHTabu as defined below.

SAHHTabu (in terms of minimisation)

r_{beg} = the acceptance ratio at the start.
γ = reduction factor for r. (In practice the finishing ratio r_{end} is specified and γ is calculated to achieve this value in the correct number of iterations.)
t = temperature
β = temperature reduction parameter; α = temperature increase parameter
k = repetitions per temperature = no. iterations per low-level heuristic application
i = iteration counter
$nits$ = total number of iterations.
$z(s)$ = cost of solution s
Initialise:
Set $r_{\text{beg}}, \beta, t, k, \gamma, nits$
$r = r_{\text{beg}}$
$\alpha = r_{\text{beg}} \cdot \beta$
Set tabu_status h = '*false*' for each low-level heusistic, h.
Set max_tabu, the length of the tabu list, to an appropriate value.
$nrep = 1$
Select a random initial solution s_0
$Cost_p = f(s_0)$
$i = 1$
While $i \leqslant 1, nits$
  Let h^* be the highest rank non-tabu low-level heuristic.
  $Cost_c = \infty$
  For $j = 1, nrep$
    $i = i + 1$
    Select s, a neighbour of s_0 according to the rules of h_*
    $Cost_c = \min\{Cost_c, z(s)\}$
    Let $\Delta = z(s_0) - z(s)$
    If $\Delta \leqslant 0$ s_0 then
      $s_0 = s$
    else
      $s_0 = s$ with probability $\exp(-\Delta/t)$
    endif
  If $s_0 = s$ then
    $t = t/(1+\beta t)$
  else

```
        t = t/(1 − αt)
    end_if
    next j
    if(mod(nits, k)) = 0 then
        r = r · γ
        α = r · β
        if nits/k = 1 then nrep = k
    endif
    δ = cost_p − cost_c
    call select low-level heuristic.
    cost_p = cost_c
end_while
```

Note that for the first k iterations the low-level heuristic is updated every move, rather than every k moves. This feature was incorporated as the number of accepted/improving moves at the start of the search is high, and the first heuristic selected is likely to achieve an artificially high score and dominate the search for some time.

6. Experiments

At 21 the total number of low-level heuristics is somewhat higher than that recommended in Soubeiga (2003). Therefore the initial plan was to use SAHHTabu with the 21 low-level heuristics to identify those that showed promise, by counting the number of times each resulted in an improvement according to measure δ. If a small set of, say 5–8, dominant heuristics could be identified then it seemed reasonable to assume that an implementation based on this subset should either produce better solutions, or converge to solutions of equal quality more quickly, as less time would be wasted on the less useful variants. However, before conducting such experiments it was necessary to determine whether or not SAHHTabu could improve over the results obtained by simple descent and to measure the influence of the cooling parameters.

Twenty runs were carried out for each of the values of p from 3 to 12 using $(r_{\text{beg}}, r_{\text{end}})$ values of $(50, 250)$, $(10, 1000)$ and $(2, 5000)$ with γ calculated in order to reduce r between these limits over 1000 sets of 20 000 iterations. These were compared with 20 000 000 iterations of the descent algorithm using N2 and the results of 2 000 000 iterations using N1 for $p = 3$, 6 and 9. It is not necessary to run N1 with additional iterations as the results show that this neighbourhood has already converged after 2 000 000 iterations. In view of the relatively poor quality of these results runs for other values of p were not executed. The results are presented in Table 2. They show that the SAHHTabu improves significantly on the same number of iterations using a pure descent method, with some evidence that slower cooling over a mid acceptance range tends to produce better results than the faster cooling variants. Typical standard deviations are now in the range 0.01–0.03, suggesting that the search consistently converges to good solutions. An analysis of the frequency with which each low-level heuristic resulted in an improvement showed that the highest scoring heuristic(s) varied from run to run, but that the smaller Nx(10) neighbourhoods featured more strongly as p increased, suggesting that the hyperheuristic was adapting to the properties of the search. Six low-level heuristics also featured consistently in the top 10. These were N1 with S3, S4 and S5, N1(10) with S1 and S2 and N2 with S5. Parameters $(50, 250)$ were used to compare the 21 low-level heuristic algorithm with the version using only these six low-level combinations. The results are given in the last column of Table 2 and show little difference in terms of solution quality. More surprisingly the latter also takes as long as the 21 heuristic variant to converge. Thus it appears that all 21 combinations may have a role to play. For example the less aggressive nature of the search when using all 21 heuristics may make it easier to diversify, which may coun-

Table 2
Results for SSHHTabu compared with pure descent. Mean utilisation over 20 runs

p	Random descent		AHHTabu			
			21 low-level heuristics			6 low-level heuristics
	N1	N2	(25 000)	(101 000)	(50 250)	(50 250)
3	88.03	88.16	89.23	89.35	**89.39**	89.38
4	–	90.68	91.44	91.48	91.44	**91.49**
5	–	92.70	93.57	93.60	**93.64**	**93.64**
6	93.65	94.10	94.83	94.83	94.92	**94.93**
7	–	95.15	95.76	95.76	**95.80**	95.76
8	–	95.98	96.51	96.53	**96.55**	96.52
9	96.29	96.66	97.17	97.21	**97.25**	97.24
10	–	97.15	97.75	**97.76**	**97.76**	97.74
11	–	97.56	98.12	98.14	**98.16**	98.10
12	–	98.06	98.39	98.45	**98.48**	98.43
No. bests	0	0	9	7	8	5

The best performance for each p is shown in bold.

ter balance slower convergence. The last row of the table shows the number of runs that converged to the best solution obtained for each value of p. Based on this measure using 6 low-level heuristics is not as good as using all 21. It was therefore decided to maintain all 21 heuristics in the final version to provide the additional flexibility.

The final version was also run on two further data sets. In the first, noise was added to the dimensions of the original data set, varying dimensions by up to $\pm 10\%$. In the second the products from the original and noised data sets were combined to form a set of 92 products. The results, together with those for the original data set are shown in Fig. 1.

The results for the new data sets are not as good as those for the real data. This is not surprising as the set of real products often have common dimensions and families of product therefore fit well in the same container. In the noised data this is not the case. The results for the combined data set are worse than the other two. Again this is to be expected as a greater number of diverse products will be allocated to each

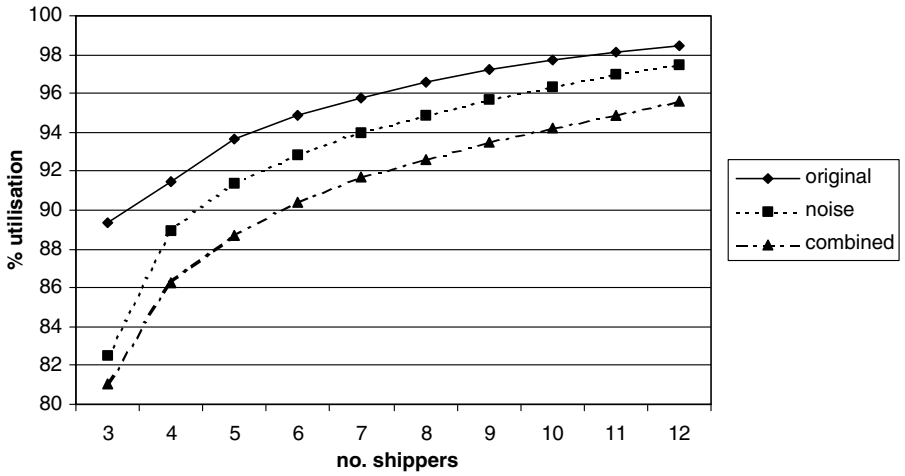

Fig. 1. Performance on three data sets.

shipper. Nevertheless all three runs show that high volumetric utilisations can be obtained and that SAHH-Tabu is an appropriate tool for analysing the effects of shipper rationalisation on volumetric utilisation, and for examining the trade-off between the number of shippers and the additional volume required. Solution times for 46 products ranged from 539.2 seconds per run when $p = 3$ to 551.2 seconds when $p = 12$ and those for 92 products from 950.8 seconds when $p = 3$ to 993.4 seconds when $p = 12$. All runs were carried out on a Pentium IV 2.4.

7. Conclusions and further research

The main focus of this work was to provide a flexible solution approach to the shipper rationalisation problem. The version of the problem used to test our approach is just one of many variants. Some changes in problem specification (for example limitations on the aspect ratios of the shipper dimensions, incorporation of cardboard thickness into the cost calculations and the relaxation of the constraint that everything must be placed 'right way up') could be dealt with using only minor changes to the current code, although this could change the characteristics of the solution landscape. Others (for example, including the palletisation efficiency of the shippers), would result in further possibilities for solution space reduction and sampling policies. The hyperheuristic approach seems a good way of incorporating such changes without the need for extensive experimentation to determine their impact on solution speed or quality.

The paper has also introduced the tabu search hyperheuristic within a simulated annealing framework. Two potential problems with applying HHTabu directly were raised and ways of overcoming this were suggested and implemented. However, these solutions themselves raise further questions. For example the decision to use the best solution, as opposed to any of the suggestions based on the mean, to calculate δ was based on a very limited set of experiments. It would be interesting to investigate this further, possibly including other potential measures. The decision to use $k = 20\,000$ was also fairly arbitrary. Further experimentation into the balance between k and the ratio $nits/k$ could be carried out. Similarly, is the undulating temperature mechanism the best way to deal with the effects of different neighbourhood sizes on the volatility at a given temperature, or might some other approach be more suitable? All these issues warrant further investigation on this and other problems.

Finally we observe that, as one of the characteristics of simulated annealing is to avoid an aggressive search for deep local optima during its early phases, the hyperheuristic policy of rewarding low-level heuristics based on low cost alone may not be the best strategy. Cowling et al. (2002) suggest more sophisticated measures but the focus of these is still on seeking out good solutions as quickly as possible. It is possible that other measures that more closely reflect the behaviour of the simulated annealing algorithm over the length of a single run may prove more appropriate. These and other issues relating to the use of a hyperheuristic approach within the framework of a simulated annealing algorithm would form an interesting area for further research.

References

Beasley, J.E., 1985. A note on solving large *p*-median problems. European Journal of Operational Research 21, 270–273.

Bischoff, E., Dowsland, W.B., 1982. The application of the micro to product design and distribution. Journal of the Operational Research Society 33, 271–280.

Burke, E.K., Kendall, G., Soubeiga, E., 2003a. A Tabu Search Hyper-Heuristic for Timetabling and Rostering. Journal of Heuristics 9 (6), 451–470.

Burke, E.K., Kendall, G., Newall, J.P., Hart, E., Ross, P., Schulenburg, S., 2003b. Hyper-heuristics: An Emerging Direction in Modern Search Technology. In: Glover, F., Kochenberger, G. (Eds.), Handbook of Meta-heuristics. Kluwer, pp. 457–474, Chapter 16.

Burke, E.K., Kendall, G., Whitwell, G., 2004. A New Placement Heuristic for the Orthogonal Stock Cutting Problem. Operations Research 52 (4), 655–671.

Cagan, J., Shimada, K., Yin, S., 2002. A survey of computational approaches to three-dimensional layout problems. Computer Aided Design 34, 597–611.

Congram, R., Potts, C., Van de Velde, S., 2002. An iterated dynasearch algorithm for the single-machine weighted tardiness problem. INFORMS Journal of Computing 14, 52–67.

Cowling, P., Kendall, G., Soubeiga, E., 2000. A hyperheuristic approach to scheduling a sales summit. In: Practice and Theory of Automated Timetabling III (PATAT 2000). In: Burke, E.K., Erben, W. (Eds.), LNCS, 2079. Springer, pp. 176–190.

Cowling, P., Kendall, G., Soubeiga, E., 2002. Hyperheuristics: A tool for rapid prototyping in scheduling and optimisation. In: Cagoni, S., Gottlieb, J., Hart, E., Middendorf, M., Günther, R. (Eds.), Applications of Evolutionary Computing: In: Proceedings of Evo Workshops 2002, Kinsale, Ireland, April 3–4, 2002, LNCS 2279, Springer–Verlag, pp. 1–10.

Crispin, A.J., Clay, P., Taylor, G.E., Bayes, T., Reedman, D., 2003. Genetic algorithms applied to leather lay plan material utilisation. IMechE Part B: Journal of Engineering Manufacture 217, 1753–1756.

Dowsland, K.A., 1985. Determining an upper bound for a class of rectangular packing problems. Computers and Operational Research 12, 201–205.

Dowsland, K.A., 1987a. An exact algorithm for the pallet loading problem. European Journal of Operational Research 31, 78–84.

Dowsland, K.A., 1987b. A combined database and algorithmic approach to the pallet loading problem. Journal of the Operational Research Society 38, 341–345.

Dowsland, K.A., 1993a. Using simulated annealing for efficient allocation of students to practical classes. In: Applied Simulated Annealing. In: Vidal, R.V.V. (Ed.), LNEMS, 396. Springer, pp. 151–174.

Dowsland, K.A., 1993b. Some experiments with simulated annealing techniques for packing problems. European Journal of Operational Research 68, 389–399.

Estivill-Castro, V., Torres-Velazquez, R., 1999. Hybrid genetic algorithm for solving the *p*-median problem. In: SEAL'98. In: Yao, X et al. (Eds.), LNCS, 1585. Springer, pp. 18–25.

Garcia-Lopez, F., Melian-Batista, B., Moreno-Perez, J.A., Moreno-Vega, J.M., 2003. Parallelization of the scatter search for the *p*-median problem. Parallel Computing 29, 575–589.

Han, L., Kendall, G., 2003. Guided Operators for a Hyper-Heuristic Genetic Algorithm. In: T.D. Gedeon, L.C.C. Fung (Eds.) Proceedings of AI-2003: Advances in Artificial Intelligence. The 16th Australian Conference on Artificial Intelligence (AI'03), Perth, Australia 3–5 December 2003. Springer Lecture Notes in Artificial Intelligence 2903, pp. 807–820.

Hanjoul, P., Peeters, D., 1985. A comparison of two dual-based procedures for solving the *p*-median problem. European Journal of Operational Research 20, 387–396.

Hansen, P., Mladenovic, N., 1997. Variable neighbourhood search for the *p*-median. Location Science 5, 207–226.

Hopper, E., Turton, B.C.H., 2001. An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem. European Journal of Operations Research 128, 34–57.

Hribar, M., Daskin, M.S., 1997. A dynamic programming heuristic for the *p*-median problem. European Journal of Operational Research 101, 499–508.

Kendall, G., Mohamad, M., 2004. Channel Assignment Optimisation Using a Hyper-heuristic. Accepted for The 2004 IEEE Conference on Cybernetics and Intelligent Systems (CIS'04), Singapore, 1–3 December.

Kendall, G., Mohd Hussain, N., 2004. A Tabu Search Hyper-heuristic Approach to the Examination Timetabling Problem at MARA University of Technology. In: Proceedings of the 5th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2004), August 18-02-2004, Pittsburgh, USA, pp. 199–217.

Leung, J.Y.-T., Dror, M., Young, G.H., 2001. A Note on an Open-end Packing Problem. Journal of Scheduling 4 (4), 201–207.

Lodi, A., Martello, S., Monaci, M., 2003. Two-dimensional packing problems: A survey. European Journal of Operations Research 141, 241–252.

Lundy, M., Mees, A., 1986. Convergence of an annealing algorithm. Maths Programming 34, 111–124.

Menon, S., Schrage, L., 2002. Order allocation for stock cutting in the paper industry. Operations Research 50 (2), 324–332.

Morabito, R., Arenales, M., 2000. Optimizing the cutting of stock plates in a furniture company. International Journal of Production Research 38 (12), 2725–2742.

Mukhacheva, E.A., Mukhacheva, A.S., 2004. The Rectangular Packing Problem: Local Optimum Search Methods Based on Block Structures. Automation and Remote Control 65 (2), 248–257.

Orlin, J., Ahuja, R., Ergun, O., Punnen, A., 2002. A survey of very large scale neighbourhood techniques. Discrete Applied Maths 23, 75–102.

Puchinger, J., Raidl, G.R., Koller, G., 2004. Solving a real-world glass cutting problem. In: Evolutionary Computation in Combinatorial Optimization. In: Gottlieb, J., Raidl, G. (Eds.), LNCS, 3004. Springer, pp. 165–176.

Resende, M.G.C., Werneck, R.F., 2004. A hybrid heuristic for the *p*-median problem. Journal of Heuristics 10, 59–88.

Rolland, E., Schilling, D.A., Current, J.R., 1996. An efficient tabu search procedure for the *p*-median problem. European Journal of Operational Research 96, 329–342.

Ross, P., Schulenburg, S., Marín-Blázquez, J.G., Hart, E., 2002. Hyper-heurstics: Learning to combine simple heuristics in bin-packing problems. In: GECCO 2002: In: Proceedings of the Genetic and Evolutionary Computation Conference, New York, NY, 2002. Morgan Kauffmann, pp. 942–948.

Ross, P., Marín-Blázquez, J.G., Schulenburg, S., Hart, E., 2003. Learning a Procedure That Can Solve Hard Bin-Packing Problems: A New GA-Based Approach to Hyper-heurstics. In: Genetic and Evolutionary Computation-GECCO 2003. In: Goos, G., Hartmanis, J., van Leewen, J. (Eds.), LNCS, 2724. Springer, pp. 1295–1306.

Soubeiga, E., 2003. Development and application of hyperheuristics to personnel scheduling. PhD Thesis, Department of Computer Science, The University of Nottingham.

Teitz, M.B., Bart, P., 1968. Heuristic methods for estimating the generalized vertex median of a weighted graph. Operations Research 16, 955–961.

Thompson, J.M., Dowsland, K.A., 1996. General cooling schedules for a simulated annealing based timetabling system. In: Practice and Theory of Automated Timetabling. In: Burke, E., Ross, P. (Eds.), LNCS, 1153. Springer, pp. 345–363.

Thompson, J.M., Dowsland, K.A., 1998. A robust simulated annealing based timetabling system. Computers and Operational Research 25, 637–648.

Whitaker, R., 1983. A fast algorithm for the greedy interchange for large scale clustering and median location problems. INFOR 21, 95–108.

Yeung, L.H.W., Tang, W.K.S., 2003. A Hybrid Genetic Approach for Garment Cutting in the Clothing Industry. IEEE Transactions on Industrial Electronics 50 (3), 449–455.