ELSEVIER

Knowledge-Based Systems 13 (2000) 159–165

Knowledge-Based SYSTEMS

www.elsevier.com/locate/knosys

Structured cases in case-based reasoning—re-using and adapting cases for time-tabling problems

E.K. Burke[a,*], B. MacCarthy[b], S. Petrovic[a], R. Qu[a]

[a]*School of Computer Science and Information Technology, The University of Nottingham, Nottingham, NG8 1BB, UK*
[b]*Division of Manufacturing Engineering and Operations Management, The University of Nottingham, Nottingham, NG7 2RD, UK*

Abstract

In this paper, we present a case-based reasoning (CBR) approach solving educational time-tabling problems. Following the basic idea behind CBR, the solutions of previously solved problems are employed to aid finding the solutions for new problems. A list of feature–value pairs is insufficient to represent all the necessary information. We show that attribute graphs can represent more information and thus can help to retrieve re-usable cases that have similar structures to the new problems. The case base is organised as a decision tree to store the attribute graphs of solved problems hierarchically. An example is given to illustrate the retrieval, re-use and adaptation of structured cases. The results from our experiments show the effectiveness of the retrieval and adaptation in the proposed method.

Keywords: Case-based reasoning; Time-tabling problems; Attribute graphs

1. Introduction

Case-based reasoning (CBR) [1] solves problems by retrieving the most similar previous cases in a case base (*source cases*) and by re-using the knowledge and experiences from previous good quality solutions. If necessary, the retrieved solutions are adapted by using domain knowledge so that they are applicable for the new problem. The case base is then updated by the new learned cases.

1.1. Traditional case representation in CBR

In traditional CBR, a list of feature–value pairs is typically employed to represent cases. The nearest-neighbour method is used extensively as a similarity measure that gives every feature a weight and results in a weighted sum to measure the similarity between two cases. Then the most similar case(s) retrieved from the case base are adapted for the new problem. In some domains, this representation and retrieval method is sufficient to find similar cases. However, some complex problems (such as time-tabling problems) consist of events that are heavily inter-connected with each other. A list of feature–value pairs by itself is not able to describe important information that could make differences in finding high quality solutions to this kind of problem. Thus the similarity measure cannot recognise the correspondence between the features in cases and characteristics of the solutions. It can be very difficult to adapt the retrieved cases for the new problems and the adaptation may take as much effort as scheduling from scratch. Smyth and Keane [2] questioned the similarity assumption in CBR and introduced a concept called “adaptation-guided retrieval”. It is unwarranted to assume that the most similar case is also the most appropriate from the re-use perspective. Similarity must be augmented by a deeper knowledge about how easy it is to modify a case to fit a new problem. Traditional case representation does not enable this description of the deeper knowledge that is needed in cases such as the heavily inter-connected time-tabling problem. A similarity measure such as the nearest-neighbour method is not sophisticated enough to reflect deeper similarities between these problems.

The aim of this paper is to present the possibilities and advantages of using attribute graphs to represent cases structurally in a CBR system which solves educational time-tabling problems. The attribute graphs are used to describe the relations between the events in a time-tabling problem more concisely and explicitly and thus can express deeper knowledge stored in the cases such as the correspondence between structures of events and characteristics of the solutions. The solutions of the retrieved cases are adaptable

* Corresponding author. Tel.: +44-1159514206; fax: +44-1159514254.

0950-7051/00/$ - see front matter
PII: S0950-7051(00)00057-5

and can be reused for the new problem that has a similar structure.

1.2. Structured cases in CBR

Representing cases structurally has been discussed in the literature, but no general theory or methodology has been identified. Böner [3] proposed a CBR system that transformed a set of pre-selected candidate cases into a structural representation to find the common structure between the new problem and candidate cases. This approach was also used to represent the topological structure to support layout design [4]. Structural similarity is usually defined using maximum common sub-graphs, which are employed as prototypes to represent classes of cases thus reducing much of the memory retrieval effort.

Racci and Sender [5] used a tree to represent structured cases. The similarity measure takes into account both the structures and the labels in the cases. A set of algorithms was explored to solve subtree-isomorphism and it was shown that significant speed-up can be obtained on randomly generated case bases.

Two systems CHIRON and CAPER were used in Ref. [6] to show how the graph-structured representation implemented as semantic networks support Case-Based Planning in two domains. The benefits and cost associated with graph-structured representation were also discussed. In CAPER, the retrieval problem was solved by a massively parallel mechanism [7].

Jantke introduced "nonstandard concepts" [8] where cases are represented as structured cases. The similarity measure thus takes structural properties into account, with the aim of making the CBR approach more flexible and expressive.

The FABEL project [9] provides more details of some existing systems that employ structured cases. The case similarities described were classified into five groups: restricted geometric relationships; graphs; semantic nets; model-based similarities and hierarchically structured similarities.

1.3. CBR in scheduling and time-tabling problems

1.3.1. CBR in scheduling

As far as the authors are aware, there are few publications specifically on CBR for scheduling problems. MacCarthy [10] proposed a general framework for CBR in general scheduling environments and the areas where CBR offered the most potential were justified. A review was also carried about CBR systems dealing with scheduling problems. Koton [11] proposed a system for the scheduling of a large-scale airlift management problem by abstracting and decomposing it, and afterwards the precedent cases were combined for the new problem. The CBR-1 project [12] used CBR in the dynamic job-shop scheduling problem. A pool of methods in the system provides rules dealing with a constrained environment but it requires a large amount of memory. Miyashita and Sycara [13] stored previous schedule repair tactics as cases in the CABIN system for job shop scheduling problems by incrementally revising a complete but sub-optimal schedule to produce a better schedule according to a set of optimisation criteria.

Hennessy and Hinkle [14] explored a new approach for retrieval and adaptation processes to solve the autoclave management and loading problem. Case adaptation finds the substitute by searching the case that has the correct context in the new environment for the unmatched parts. In Ref. [15] two approaches were explored that reuse the portions of good schedules to build new schedules. The experiment results were compared with other methods and showed that the approach worked efficiently for less-complex scheduling problems. Schmidt [16] proposed a problem solving system that used the theory of scheduling within CBR to solve production planning and control problems. Scheduling problems are organised by using "transformation graphs" to show similarities between problem characteristics in terms of polynomial transformation between cases. MacCarthy and Jou [17] discussed the use of CBR in the development of a class of scheduling problems involving sequence dependent set-up times. General problems about the application of CBR to scheduling problems were also addressed.

1.3.2. Time-tabling problems

In this paper, structured cases are used in CBR to represent simple educational time-tabling problems. Time-tabling problems were defined by Wren [18] as: "the allocation, subject to constraints, of given resources to objects being placed in space-time, in such a way as to satisfy as nearly as possible a set of desirable objectives." Time-tabling problems are specific types of scheduling problems that can be highly constrained and difficult to solve. A general time-tabling problem consists of assigning a number of events (e.g. exams, courses, meetings, etc.) into a limited number of timeslots (periods of time) so that no person is assigned to two or more events simultaneously. Constraints which should under no circumstances be violated are known as hard constraints. Other constraints which are desirable but not essential (such as that two events should be consecutive, etc.) are known as soft constraints. The violations of the soft constraints should be minimised.

Various methods have been used to solve educational time-tabling problems [19,20]. The graph theoretic approach was a widely employed technique in the early days of research on time-tabling problems.

Recent research has considered a variety of modern metaheuristic methods and approaches such as Tabu Search (e.g. see Refs. [21,22]), Simulated Annealing (e.g. see Refs. [23,24]), Genetic Algorithms (e.g. see Refs. [25–27]) and hybrid methods (such as Memetic Algorithms, e.g. see Refs. [28–32]). A wide variety of research work on time-tabling can be found in Refs. [33,34].

CBR is potentially a very valuable tool in scheduling

Table 1
Some node attributes of course time-tabling problem

Label	Attribute	Value(s)	Notes
0	Ordinary course	N/A	Takes place once a week
1	Multiple course	*N* (no. of times)	Takes place *N* times a week
2	Pre-fixed course	*S* (slot no.)	Assigned to timeslot *S*
3	Exclusive course	*S* (slot no.)	Not assigned to timeslot *S*

problems [10,16]. One of the major contributions of CBR, as a modelling tool to capture knowledge, is the ability of avoiding computation (in Tabu search, GA, etc.) by searching, selection and matching techniques. CBR is also a valuable method for scheduling problems that put emphasis on constraint directed research, a major feature in time-tabling problems.

In this paper a simple course time-tabling problem is used as an example to interpret the retrieval and adaptation of structured cases. The representation of time-tabling problems by attribute graphs is given in Section 2. Section 3 describes the implementation of the proposed system organised as a tree and an example is shown in Section 4. A brief concluding discussion is presented in Section 5.

2. Attribute graphs for course time-tabling problems

In course time-tabling, a number of courses (events) have to be assigned to a limited number of timeslots. Two courses may have common students so they conflict with each other and cannot be assigned to the same timeslot.

Attribute graphs are used here to represent course time-tabling problems structurally. In attribute graphs, nodes indicate events and edges show the relation between any pair of events. Nodes and edges have attributes that represent the problem more precisely. Each attribute corresponds to a label assigned to nodes and edges. Tables 1 and 2 show parts of the labels and attributes of nodes and edges that are used in our problems.

A simple example is shown in Fig. 1 to illustrate a course time-tabling problem represented as an attribute graph. Nodes represent courses. Solid edges indicate hard constraints (labelled 7) which means that the adjacent courses cannot be held simultaneously. Dotted lines indicate soft constraints labelled 4, 5 or 6. The labels on the edges and inside the nodes correspond to the attributes shown in Tables 1 and 2. For example, Maths, Physics and Chemistry are labelled with a 1 (to indicate that they are multiple courses) and with values 2, 3 and 2 that denote that they should be held 2, 3 and 2 times a week, respectively. Other courses are labelled 0 (ordinary courses), which denote that they should be held just once a week. SpanishA should not be consecutive to Physics (because the edge between them is labelled by a 6) and Chemistry should be consecutive to SpanishB (labelled by a 5). The directed line between SpanishA and SpanishB has the label 4 (with value 1) which denotes that SpanishA should be held before SpanishB.

Using this approach, the course time-tabling problems can be represented structurally. It enables us to describe the relations between events in the problem that is not possible by using feature–value pairs. Also the different cases of the problems can have different structures, unlike in traditional case representation using the list of feature–values pairs where all the cases have the same form of feature slots.

3. Implementation of the CBR system for course time-tabling problems

3.1. The graph isomorphism problem and decision tree algorithm

Using attribute graphs to represent cases has many advantages. However, the matching problem between the structured cases is equivalent to that of the graph isomorphism or sub-graph isomorphism problem that is known to be NP-Complete [35]. A graph G is isomorphic to graph G' if there exists a one to one correspondence between nodes and edges of the two graphs. A graph G is sub-graph isomorphic to graph G' if G is isomorphic to a sub-graph of G'. Some methods have been attempted to solve this problem in CBR by detecting cliques of the graph [3]. The system being proposed here is based on Messmer's algorithm [36] where graphs are organised in a decision tree.

The attribute graph is represented by its adjacency matrix $M = m_{i,j}$, where $m_{i,j} \in L_e$ indicates the attribute of the edge between node i and node j and $m_{i,i} \in L_n$ indicates the attribute of node i. L_e and L_n are the sets of labels defined in Tables 1 and 2. There are $n!$ different adjacency matrices for an n-node attribute graph when the nodes are in different permutations. The basic idea of Messmer's algorithm is to pre-store all the adjacency matrices of some known graphs with their permutation matrices $P = p_{i,j}$, to the corresponding nodes in a decision tree. If graph G is

Table 2
Some edge attributes of course time-tabling problem

Label	Attribute	Values(s)	Notes
4	Before/after	1 or 0 (direction)	One before/after another course
5	Consecutive	N/A	Be consecutive with each other
6	Non-consecutive	N/A	Not consecutive with each other
7	Conflict	N/A	Conflict with each other

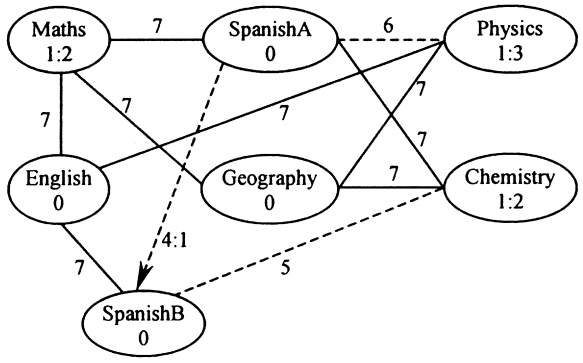

Fig. 1. Attribute graph of a course time-tabling problem.

isomorphic to graph G, then if $p_{i,j} = 1$, node i in graph G corresponds to node j in graph G'. If a new graph can be classified to a node in the decision tree at level k, then the permutation matrix (matrices) stored in this node indicate the matching between the k nodes of the new graph and that of previously stored graph(s). If the time spent on building up the decision tree is ignored, this algorithm guarantees that all the graph isomorphism(s) or sub-graph isomorphism(s) stored in the tree can be found in polynomial time (quadratic to the number of nodes of the new graph).

For example, in Fig. 2, attribute graph G represents a 3-course time-tabling problem. Maths is labelled 1 with value 2 (multiple course, held twice a week). Physics and Spanish are labelled 0 (ordinary course, held once a week). Physics should be held before Maths. Spanish should not be scheduled simultaneously with Physics as Maths. There are six adjacency matrices $M0$–$M5$ representing graph G, X denotes that there is no edge between two nodes and the labels in the matrices are described in Tables 1 and 2.

These matrices are used to build the decision tree (see Fig. 3). If a matrix M can be seen as consisting of an array of the so-called row-column elements $a_i = (m_{1i}, m_{2i}, \ldots, m_{ii}, m_{i(i-1)}, \ldots, m_{i1})$, then a 3×3 matrix consists of three elements: $a_1 = a_{11}$, $a_2 = a_{21}a_{22}a_{12}$ and $a_3 = a_{31}a_{32}a_{33}a_{23}a_{13}$. The first element of each of the matrices $M0$–$M5$ can be 1 or 0, and therefore there are two branches from the root node with label 0 and 1 on the first level. The second level under branch 1 can be 707 and $40\times$ in $M4$ and $M5$, thus two branches below branch 1 are built. Then the following levels of the decision tree can be built by the same process, each branch on level i leads to a successor node that is associated with a specific value for the ith element of $M0$–$M5$. Each permutation matrix is stored in the corresponding node in the decision tree (not shown in Fig. 3). Then all the other known attribute graphs can be added into the tree in the same way.

Let us suppose that we are presented with a new problem represented by matrix M for attribute graph G' (see Fig. 4). The matrix M is inserted into the tree and can be classified to node X according to the values of each branch. The permutation stored to node X gives the isomorphism that tells us that Maths(c), Physics(b) and Spanish(a) in attribute graph G correspond to English(b), Chemistry(a) and Maths(c) in attribute graph G', respectively.

3.2. Retrieving structurally similar cases

Some course time-tabling problems are generated randomly and their attribute graphs are used to build up a decision tree in the proposed system. The solutions of these problems are obtained by using a heuristic graph colouring method described in Ref. [37].

A penalty is associated to each pair of labels described in Tables 1 and 2 and used in the retrieval process. A threshold is also set to judge whether two labels are similar or not. When the system tries to match each pair of events in the new problem with source cases, the events can be seen as similar if the penalty between their labels is below the threshold. They are identified as similar and returned to be matched to each other.

If an event in the new problem has the same label and the same value as the source case, then they match with no penalty. Two events are considered not to be matching if the penalty between their labels exceeds the given threshold.

Two events that are labelled the same are further analysed to see if they have the same values. Penalties are given for

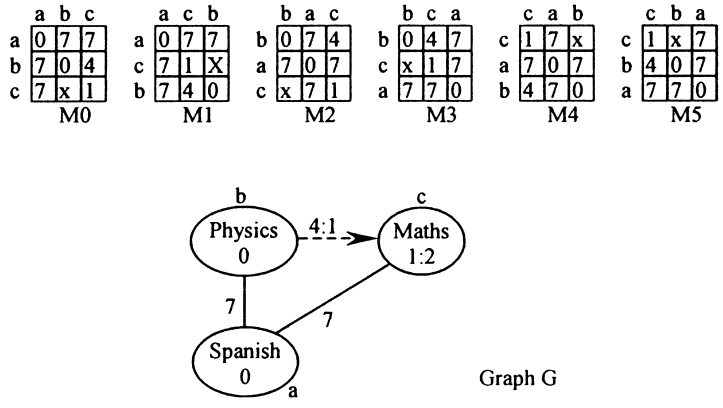

Fig. 2. Matrices of attribute graph G of a course time-tabling problem.

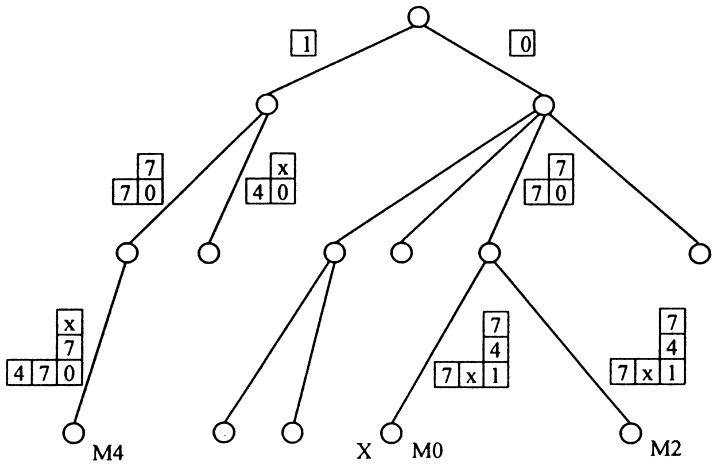

Fig. 3. Part of a decision tree storing matrices of attribute graph G of a course time-tabling problem ($M0$, $M2$ and $M4$ are shown).

the differences between the values and are taken into account in the similarity measure.

Every label is also given a weight using domain knowledge for the similarity measure. The similarity measure is thus given by

$$S = 1 - \sum_{i,j=0}^{n} p_{ij} \times w_j / P$$

where n is the total number of the labels; $p_{i,j}$ the penalty between label i of node or edge in the new problem and label j of node or edge of source cases; w_i the weight of label i in the new problem and P the sum of the penalty for every pair of labels times the weight of every label.

Using the penalties assigned to each pair of labels in the course time-tabling problems, the retrieval is targeted at matching between every pair of events, not just a single judgement between the whole cases. The system can retrieve the case(s) suitable for adaptation for the new problem from the case base.

When a new problem is entered in the system, it is classified to a node in the decision tree and the system retrieves all the cases stored in and below that node as candidates. As the tree stores cases hierarchically, all the cases that have more events and/or more relations are stored below those having less events and/or relations. It is observed that solutions of more constrained cases can be adapted easily for less constrained problems. Thus all the cases in and below the node are retrieved.

Using the penalties for every pair of the labels of nodes and edges, the system calculates the similarity between the new problem and the candidate cases in and below the node. The most similar case(s) are selected for adaptation.

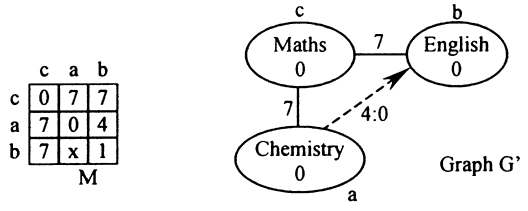

Fig. 4. Matrices of attribute graph G' for a new course time-tabling problem.

3.3. Reuse and adaptation of the solutions

After the system finds the most similar case(s), the solutions or part of the solutions of the retrieved case(s) can be reused. The system substitutes the events in the solution(s) of the retrieved case(s) with the matching events in the new problem according to the isomorphism(s) found. After the substitution, a partial solution for the new problem can be obtained although there may be some violations of constraints. If there is no violation of hard constraint in the retrieved solutions, there is also no violation of hard constraint in the solutions after substitution.

The graph heuristic method which tries to minimise the violations of constraints is used in the adaptation process. Events that violate the constraints are collected from the partial solution, and all the unscheduled events are ordered first by their degrees (number of conflicts of an event with other events) decreasingly and then are assigned one by one to the first available timeslot. If some events cannot be assigned to a timeslot without violation of constraints, they will be kept until all the other events have been scheduled. Then they are scheduled to the timeslots that lead to the fewest number of violations of constraints.

4. A simple illustrative example

Let us suppose that the problem shown in Fig. 1 is the new problem. All the cases and their isomorphisms are retrieved from the node that the new problem is classified to in the case base. Not only the case(s) that are graph isomorphic to the new problem can be adapted, but also the case(s) which the new problem is sub-graph isomorphic can be adapted, although they may not be “good” solutions for the new problem. Two cases whose similarities pass a

Table 3
Solutions after substitution by using isomorphism

	Timeslot1	Timeslot2	Timeslot3	Timeslot4	Timeslot5
Solution1	Physics, Maths, Chemistry	English, Geography	SpanishA	SpanishB, Physics, Maths	Chemistry, Maths
Solution2	Maths, Physics, Chemistry	English, Geography	SpanishA	SpanishB, Maths, Physics	SpanishB, Physics
Solution3	Physics, Maths, Chemistry	English, Geography	SpanishA	SpanishB, Physics, Maths	Chemistry, Maths
Solution4	Maths, Physics, Chemistry	English, Geography	SpanishA	SpanishB, Maths, Physics	Chemistry, Physics

given threshold (a score we set) are considered to be the most similar to the new problem and are retrieved from the case base. The structures of these two cases are shown in Fig. 5. It is possible to find more than one isomorphism between two graphs. Two isomorphisms were found for each of the retrieved cases in this example.

After substituting the events of the retrieved cases shown in Fig. 4 by matching events indicated by the isomorphisms, four solutions can be obtained for the new problem (see Table 3).

It can be seen that there are three violations of soft constraints in solution 1: SpanishA is consecutive to Physics, Physics is held only two times and Maths is scheduled one more time. Using the graph heuristic method takes two adaptation steps: it deletes Maths from timeslot1 and adds another Physics to timeslot 5. It can also be seen that there are 1, 3 and 1 violation(s) of soft constraints in solution 2, 3 and 4, respectively. Using the graph heuristic method takes 1 and 2 adaptation step(s), respectively, for solution 2 and 3. There is no adaptation for solution 4. After adaptation, there is only one violation of a soft constraint in each solution.

The simple example has demonstrated that only a few adaptations are needed to get solutions for the new problem on the basis of the solutions of the retrieved similar cases. Cases can explore deeper knowledge in course time-tabling problems by the structural representation. Retrieval that targets the adaptability of every pair of events between the new problem and the retrieved case(s) finds the most adaptable cases for the new problem, thus a corresponding relation between the events and adaptation requirements is built-up. Employing the adaptation requirements in the definition of the similarity between every event pair gives a more elaborate description for the similarity measure. Thus the knowledge and experiences previously stored in the retrieved cases' solutions can be exploited for re-use for new similar problems. We note that the CBR can re-use the sub-solutions of previously solved problems within the case-base, a manner similar to that of experts in time-tabling.

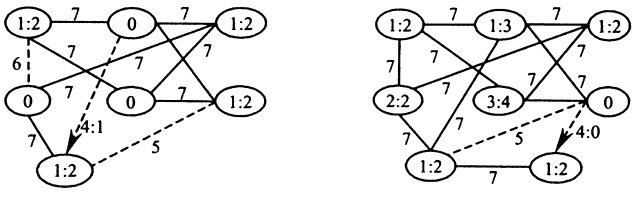

Fig. 5. Two retrieved cases from case base.

5. Conclusions and future work

In this paper, a method is proposed to help solve course time-tabling problems using CBR in which attribute graphs are used to represent cases. To our knowledge, the CBR approach proposed in this paper is new in solving the time-tabling problems. *Retrieval targets every pair of nodes and edges between the new problem and source cases so that the retrieved case(s) are the most adaptable for the new problem.* The retrieved cases' solutions store good optimised or sub-optimised schedules for the previously solved problems. These schedules can be exploited and re-used for the new similar cases, after only limited adaptations for solutions which are then applicable for the new problem. The graph data structure gives a detailed description of the time-tabling problem. The relations between any events can be described clearly, and therefore the application of *this method to time-tabling problems is likely to find the similar cases adaptable for the new problem.*

In the current system, it is presumed that some pre-compiled cases exist so that the new problems can find isomorphic or sub-graph isomorphic cases from the case base. If only part of the structure of the new problem found correspondence to part of the structures of the source cases, the partial matching could also be reused for the new problem. Research work is being undertaken on searching and re-using the maximum-weighted common sub-graphs between source cases and the new problem. This work on the simple problem is a potential method and it provides promise. We believe it is applicable to large real world problems. Comparison with other methods will form a major part of our future work in this area.

References

[1] I. Waston, F. Marir, Case-based reasoning: a review, The Knowledge Engineering Review 9 (1994) 327–354.
[2] B. Smyth, M.L. Keane, Adaptation-guided retrieval: questioning the similarity assumption in reasoning, Artificial Intelligence 102 (1998) 249–293.
[3] K. Börner, Structural similarity as guidance in case-based design, in: S. Wess, K.D. Althoff, M. Richter (Eds.), Topics in Case-based Reasoning, Springer, Kaiserslautern, 1993, pp. 197–208 (EWCBR-93).
[4] K. Börner, C.H. Coulon, E. Pippig, E.C. Tammer, Structural similarity and adaptation, in: I. Smith, B. Faltings (Eds.), Advances in Case-based Reasoning, Springer, Switzerland, 1996, pp. 58–75 (EWCBR-96).

[5] F. Ricci, L. Senter, Structured cases, trees and efficient retrieval, in: B. Smyth, P. Cunningham (Eds.), Advances in Case-based Reasoning, Selected papers from the 4th European Workshop, Ewcbr-98, Dublin, Ireland, Lecture Notes in Computer Science, Vol. 1488, Springer-Verlag, 1998, pp. 88–99.
[6] K.E. Sanders, B.P. Kettler, J.A. Hendler, The case for graph-structured representations, in: D. Leake, E. Plaza (Eds.), Case-based Reasoning: Research and Development, Selected papers from the Second International Conference, Iccbr-97, Providence, Ri, Usa, Lecture Notes in Artificial Intelligence, Vol. 1266, Springer-Verlag, 1997, pp. 245–254..
[7] W.A. Andersen, M.P. Evett, B. Kettler, J. Hendler, Massively parallel support for case-based planning, IEEE Expert 7 (1994) 8–14.
[8] K.P. Jantke, Nonstandard concepts of similarity in case-based reasoning, in: Proceedings of the 17th Annual Conference of the Geselllschaft für klassifikation e.V., Springer, Kaiderslautern, 1993.
[9] F. Gebhardt, Methods and systems for case retrieval exploiting the case structure, FABEL-Report 39, GMD, Sankt Augustin, 1995.
[10] B. MacCarthy, P. Jou, Case-based reasoning in scheduling, in: M.K. Khan, C.S. Wright (Eds.), Proceedings of the Symposium on Advanced Manufacturing Processes, Systems and Techniques (AMPST96), MEP Publications, 1996, pp. 211–218.
[11] P. Koton, SMARTlan: a case-based resource allocation and scheduling system, in: Proceedings of the Workshop on Case-based Reasoning (DARPA), 1989, pp. 285–289.
[12] A. Bezirgan, A case-based approach to scheduling constraints, in: J. Dorn, K.A. Froeschl (Eds.), Scheduling of Production Processes, Ellis Horwood, Chichester, UK, 1993, pp. 48–60.
[13] K. Miyashita, K. Sycara, Adaptive case-based control of scheduling revision, in: M. Zweben, M.S. Fox (Eds.), Intelligent Scheduling, Morgan Kaufmann, Los Altos, CA, 1994, pp. 291–308.
[14] D. Hennessy, D. Hinkle, Applying case-based reasoning to autoclave loading, IEEE Expert 7 (1992) 21–26.
[15] P. Cunningham, B. Smyth, Case-based reasoning in scheduling: reusing solution components, The International Journal of Production Research 35 (1997) 2947–2961.
[16] G. Schmidt, Case-based reasoning for production scheduling, International Journal of Production Economics 56–57 (1998) 537–546.
[17] B. MacCarthy, P. Jou, A case-based expert system for scheduling problems with sequence dependent set up times, in: R.A. Adey, G. Rzevski (Eds.), Applications of Artificial Intelligence in Engineering X, Computational Machines Publications, Southampton, 1995, pp. 89–96.
[18] A. Wren, Scheduling timetabling and rostering—a special relationship, in: E. Burke, P. Ross (Eds.), The Practice and Theory of Automated Timetabling: Selected Papers from the First International Conference, Lecture Notes in Computer Science, vol. 1153, Springer, Berlin, 1996, pp. 46–76.
[19] M.W. Carter, G. Laporte, Recent developments in practical examination timetabling, in: E. Burke, P. Ross (Eds.), The Practice and Theory of Automated Timetabling: Selected Papers from the First International Conference, Lecture Notes in Computer Science, vol. 1153, Springer, Berlin, 1996, pp. 3–21.
[20] M.W. Carter, G. Laporte, Recent developments in practical course timetabling, in: E. Burke, M. Carter (Eds.), The Practice and Theory of Automated Timetabling: Selected Papers from the Second International Conference, Lecture Notes in Computer Science, vol. 1408, Springer, Berlin, 1997, pp. 3–19.
[21] J.P. Boufflet, S. Negre, Three methods to solve an examination timetable problem, in: E. Burke, P. Ross (Eds.), The Practice and Theory of Automated Timetabling: Selected Papers from the First International Conference, Lecture Notes in Computer Science, vol. 1153, Springer, Berlin, 1996, pp. 327–344.
[22] K.A. Dowsland, Off-the-peg or made to measure? Timetabling and scheduling with SA and TS, in: E. Burke, M. Carter (Eds.), The Practice and Theory of Automated Timetabling: Selected Papers from the Second International Conference, Lecture Notes in Computer Science, vol. 1408, Springer, Berlin, 1997, pp. 37–52.
[23] J.M. Thomson, K.A. Dowsland, General cooling schedules for a simulate annealing based timetabling system, in: E. Burke, P. Ross (Eds.), The Practice and Theory of Automated Timetabling: Selected Papers from the First International Conference, Lecture Notes in Computer Science, vol. 1153, Springer, Berlin, 1996, pp. 345–364.
[24] M.A.S. Elmohamed, P. Coddington, G. Fox, A comparison of annealing techniques for academic course timetabling, in: E. Burke, M. Carter (Eds.), The Practice and Theory of Automated Timetabling: Selected Papers from the Second International Conference, Lecture Notes in Computer Science, vol. 1408, Springer, Berlin, 1997, pp. 92–112.
[25] D.C. Rich, A smart genetic algorithm for university timetabling, in: E. Burke, P. Ross (Eds.), The Practice and Theory of Automated Timetabling: Selected Papers from the First International Conference, Lecture Notes in Computer Science, vol. 1153, Springer, Berlin, 1996, pp. 181–197.
[26] W. Erben, J. Keppler, A genetic algorithm solving a weekly course-timetabling problem, in: E. Burke, P. Ross (Eds.), The Practice and Theory of Automated Timetabling: Selected Papers from the First International Conference, Lecture Notes in Computer Science, vol. 1153, Springer, Berlin, 1996, pp. 198–211.
[27] P. Ross, E. Hart, D. Corne, Some observations about GA-based exam timetabling, in: E. Burke, M. Carter (Eds.), The Practice and Theory of Automated Timetabling: Selected Papers from the Second International Conference, Lecture Notes in Computer Science, vol. 1408, Springer, Berlin, 1997, pp. 115–129.
[28] E.K. Burke, J.P. Newell, R.F. Weare, A memetic algorithm for university exam timetabling, in: E. Burke, P. Ross (Eds.), The Practice and Theory of Automated Timetabling: Selected Papers from the First International Conference, Lecture Notes in Computer Science, vol. 1153, Springer, Berlin, 1996, pp. 241–250.
[29] E.K. Burke, J.P. Newall, A multi-stage evolutionary algorithm for the timetable problem, IEEE Transactions on Evolutionary Computation 3 (1999) 63–74.
[30] E.K. Burke, J.P. Newell, R.F. Weare, Initialisation strategies and diversity in evolutionary timetabling, Evolutionary Computation Journal 6 (1998) 81–103 (Special Issue on Scheduling).
[31] B. Paechter, A. Cumming, H. Luchian, The use of local search suggestion lists for improving the solution of timetable problems with evolutionary algorithms, in: G. Goos, J. Hartmanis, J. Leeuwen (Eds.), AISB Workshop, Lecture Notes in Computer Science, vol. 993, Springer, Sheffield, 1995, pp. 86–93.
[32] B. Paechter, A. Cumming, M.G. Norman, H. Luchian, Extensions to a memetic timetabing system, in: E. Burke, P. Ross (Eds.), The Practice and Theory of Automated Timetabling: Selected Papers from the First International Conference, Lecture Notes in Computer Science, vol. 1153, Springer, Berlin, 1996, pp. 251–265.
[33] E. Burke, P. Ross (Eds.), The Practice and Theory of Automated Timetabling: Selected Papers from the First International Conference Lecture Notes in Computer Science, vol. 1153, Springer, Berlin, 1996.
[34] E. Burke, M. Carter (Eds.), The Practice and Theory of Automated Timetabling: Selected Papers from the Second International Conference Lecture Notes in Computer Science, vol. 1408, Springer, Berlin, 1997.
[35] M.R. Garey, D.S. Johnson, Computers and Intractability: a Guide to the Theory of NP-Completeness, Freeman, New York, 1979.
[36] B.T. Messmer, Efficient graph matching algorithms for preprocessed model graph, PhD thesis, University of Bern, Switzerland, 1995.
[37] E.K. Burke, D.G. Elliman, R.F. Weare, A university timetabling system based on graph colouring and constraint manipulation, Journal of Research on Computing in Education 27 (1994) 1–18.