STAIRS 2002
STarting Artificial Intelligence Researches Symposium, July 22-23 2002, Lyon, France
T.Vidal and P.Liberatore (Eds.)
IOS press, 2002

A Tabu Search Approach for Graph-Structured Case Retrieval

Sanja PETROVIC, Graham KENDALL, Yong YANG
E-mail: {sxp, gxk, yxy}@cs.nott.ac.uk
School of Computer Science and Information Technology, The University of Nottingham, Nottingham NG8 1BB, UK

Abstract. In case-based reasoning (CBR), graph-structured representations are desirable for complex application domains such as planning and design. Graph is a powerful data structure and allows knowledge to be encoded completely and expressively. However, the advantages come with a computational overhead for case retrieval, which presently prevents the usage of graph-structured representation for large-scale problems. In this paper, we describe a two-stage strategy that is based on the tabu search for solving the graph-structured case retrieval. The experiments were performed on synthetically generated graphs. The preliminary results obtained show the effectiveness of our approach.

Keywords: case-based reasoning, tabu search, structured-case representations, retrieval, graph matching.

1 Introduction

Case Based Reasoning (CBR) [1] is a problem solving methodology [2], which is applicable to a variety of domains. A CBR system represents problem solving experience as cases and retains them in the case base. It solves new problems by using solutions to old cases and adapting them to meet the requirements of the new problems [3]. The process usually consists of four main activities: retrieve the similar cases, reuse the solutions, revise the solution, and retain the past experience as a new case [4].

Case retrieval is the first step and consists of two phases: search function for identifying a set of potential cases; and case matching, which is a more comprehensive similarity evaluation for ranking cases in the collected set [1]. Case representation and case retrieval are critical for CBR, as the performance of CBR systems depends on whether knowledge can be encoded completely and properly, and whether cases whose solutions are relevant to the new problem can be retrieved [5].

1.1 Graph-Structured Representation

Case representation methods can be classified into feature-based representation and graph-structured representation [6]. In the feature-based representation, cases are represented as fixed sets of attribute-value pairs. This method is widely used by many traditional CBR systems, because cases can be retrieved efficiently in terms of time and space. A number of good search functions are available, such as nearest neighbour, induction and template

retrieval [7]. Case matching is simply implemented by summing the similarity for each attribute value pair. However, the feature-based representation is not suitable to be adopted in complex application domains where inherent structures exist in the case content and must be encoded completely and expressively.

The beauty of the graph-structured representation is its power to show the inherent structures for cases in complex application domains such as university timetabling [8], transport logistics [9], tax planning [10], academic tutoring [11], design [12, 13], and scheduling [14]. Graphs used by present CBR systems include attribute graphs (directed labelled graphs), semantic nets and conceptual graphs. In this research, we consider directed labelled graphs as the other two graph types can be transformed into directed labelled graphs. In graph representation, case's objects can be represented as vertices, and relations between objects can be represented as arcs. The labels correspond to attribute descriptors for vertices and arcs, and usually are unary and binary predicates. We refer to [6] for further readings about the advantages rendered by the graph-structured representation.

1.2 Retrieval of Graph-Structured Cases

As cases are represented as graphs, the case retrieval is equivalent to a graph matching problem, or more precisely to a sub-graph isomorphism detection problem. This problem has been shown to be NP-complete [15]. As a result, one disadvantage of using graph representation resides in the computational complexity of case matching, which increases exponentially with the number of vertices. Moreover, another bottleneck problem is the search function that is used to find the potential similar subset of graph-structured cases in the whole case base. To the authors' knowledge, no efficient general retrieval algorithms have been proposed for problems of large sizes. Our overall goal in this paper is to render a fast and robust graph-structured case retrieval.

In the field of CBR, some researchers have put their efforts into case retrieval, where cases are represented as directed labelled graphs. Bunke and Messmer [16] transform one graph into another graph for case matching. The transformation procedure is implemented by a set of operators, such as insertion, deletion and substitution of a vertex, a label, or an arc. The similarity degree for two graphs is evaluated by the number of operators required for implementing the transformation, but the problem still remains NP-Complete. Coulon [13] solves case matching by creating a new graph that is a combination of the two matched graphs and detecting the maximum cliques (the completely connected sub-graphs), but the problem of the maximum clique detection is also NP-complete [15]. Further work by Messmer and Bunke [17] implements graph retrieval efficiently by decomposing graphs into subgraphs and organising these subgraphs into a decision tree. However, the efficient retrieval time comes at the cost of exponential storage space, so their approach is only applicable to small graphs. Wang and Ishil [18] shed some light on the issue by exploiting a general similarity measure for graph-structured CBR systems. They also developed a genetic algorithm for case matching.

It is worth reviewing some relevant works in the literature of graph matching, where graphs do not have attribute descriptors (labels) for vertices or arcs. This problem has been studied for over 30 years [19]. Various state-space search approaches were reported including exact subgraph isomorphism [20] and heuristic techniques [21]. Domain-specific heuristic methods have been neglected in the literature for a period due to the development of modern metaheuristic methods [22] such as simulated annealing [23], genetic algorithms [24] and memetic algorithms [25]. In [22], Williams and Wilson implement a heuristic method for image recognition, which is somewhat similar to tabu search. They firstly find a

predefined number of vertex maps that have the highest probability of clique match consistency by randomly establishing graph vertex maps for several times. These vertex-maps are kept as the unchanged components (seed) in the solution during the search. Then the search is focused on the neighbours of the seed. Any good vertex maps found in the search process will also be kept as parts of seed.

In this paper, we propose a tabu search [26] approach for graph-structured case retrieval, which employs domain specific knowledge of directed labelled graphs. We use tabu search because domain specific knowledge can be added into the components in tabu search relatively easily (see Section 3). This paper is organised as follows. The similarity measure is introduced in Section 2. In Section 3, we describe our implementation of tabu search. In Section 4, a two stage graph-structured case retrieval strategy is proposed. In section 5, experimental results are reported and analysed. Conclusions are discussed in Section 6.

2. Similarity Measure

We first give the basic definition of the directed-labelled graph adopted from [17,18].

> Definition 1 The directed-labelled graph is denoted by $G = (V, A, \alpha, \beta, L_v, L_a)$, where V is the set of vertices, $A \subseteq V \times V$ is the finite set of arcs, $\alpha: V \mapsto L_v$ is the vertex labeling function, $\beta: V \mapsto L_a$ is the arc labeling function, L_v is the finite set of the symbolic labels for graph vertices, L_a is the finite set of the symbolic labels for graph arcs. The weight of vth vertex and the weight of ath arc are denoted by $W(v)$ and $W(a)$ respectively, and should be decided beforehand.

In this paper, we use $|V|$ to denote the cardinality number of a set V. We assume that the numbers of $|L_v|$ and $|L_a|$ are much smaller than those of $|V|$ and $|A|$ respectively, which means that the same labels will be set on many different vertices and arcs. We make this assumption, because it is a chief feature of graph-structured CBR systems in the real-world applications.

The target graph representing a graph to be matched and the source graph from the case base are denoted by $G_t = (V_t, A_t, \alpha_t, \beta_t, L_{vt}, L_{at})$ and by $G_s = (V_s, A_s, \alpha_s, \beta_s, L_{vs}, L_{as})$, respectively. In order to compute the similarity between G_t and G_s, it is sufficient to establish a vertex-to-vertex correspondence that associates vertices in G_t with those in G_s, as mapping process of arcs depends on the map of vertices. A correspondence from graph G_t to G_s is represented by a function $f: V_t \rightarrow V_s$. The function f defines a set of Cartesian pairs in the space of all possible maps between G_t and G_s, i.e. $f \subseteq V_t \times V_s$. A vertex correspondence from a vertex $u \in V_t$ to a vertex $v \in V_s$ in G_s is denoted by $f(u) = v$. An arc correspondence from an arc $x \in A_t$ to an arc $y \in A_s$ is denoted by $f(x) = y$. In this paper, we adopt the definitions for the similarity degree of mapped vertex and arc pairs proposed by Wang [18]. They are described briefly as follows, as they will be used throughout the paper. More detailed explanations and an example for computing similarity degree are given in [18].

Definition 2 The similarity degree between vertices $u \in V_t$ and $v \in V_s$ in correspondence f is denoted by $DS_f(u, v)$. If $f(u) \neq v$, then $DS_f(u, v) = 0$, else

$$DS_f(u, v) = \begin{cases} 1, & \text{If } \alpha_t(u) = \alpha_s(v) \\ 0, & \text{If } \alpha_t(u) \neq \alpha_s(v) \end{cases} \quad (1)$$

Definition 3 If $n_{x1}, n_{x2} \in V_t$ and $n_{y1}, n_{y2} \in V_s$, the similarity degree between arc $x = (n_{x1}, n_{x2})$ and $y = (n_{y1}, n_{y2})$ in correspondence f is denoted by $DS_f(x, y)$. If $f(x) \neq y$, $DS_f(x, y) = 0$, else

$$DS_f(x, y) = \begin{cases} 1, & \text{If } f(n_{x1}) = n_{y1} \text{ and } f(n_{x2}) = n_{y2} \text{ and } \beta_t(x) = \beta_s(y) \\ 0.5, & \text{If } f(n_{x1}) = n_{y1} \text{ and } f(n_{x2}) = n_{y2} \text{ and } \beta_t(x) \neq \beta_s(y) \\ 0, & \text{If } f(n_{x1}) \neq n_{y1} \text{ or } f(n_{x2}) \neq n_{y2} \end{cases} \quad (2)$$

Definition 4 Label ϕ is used to denote an extraneous vertex or arc in a graph. Let $DS_f(u, \phi) = 0$, $DS_f(\phi, v) = 0$, $DS_f(x, \phi) = 0$, $DS_f(\phi, y) = 0$, $W(\phi) = 0$.

Definition 5 The similarity degree between G_t and G_s in correspondence f is denoted by $DS_f(G_t, G_s) \in [0, 1]$,

$$DS_f(G_t, G_s) = \frac{F_n + F_e}{M_n + M_e} \quad (3)$$

where

$$F_n = \sum_{n_t \in V_t} \frac{W(n_t) + W(f(n_t))}{2} DS_f(n_t, f(n_t)) \quad (4)$$

$$F_e = \sum_{e_t \in A_t} \frac{W(e_t) + W(f(e_t))}{2} DS_f(e_t, f(e_t)) \quad (5)$$

The values of F_n and F_e are the sum of similarity degrees for all mapped vertex and arc pairs, respectively. M_n and M_e are the maximum value of F_n and F_e, respectively. The similarity degree between two graphs could be computed differently by different correspondences. Therefore the problem of graph-structured case retrieval can be transformed into the search for the best correspondence between two graphs, which renders the largest similarity degree. We develop a tabu search algorithm to search for the best correspondence (the solution of the tabu search) between the target case and the cases in the case base.

3. Tabu Search

Tabu search is a meta-heuristic that guides a heuristic search to explore the solution space beyond local optimality [26]. The chief feature of tabu search is an effective use of the adaptive memory. The memory can be either short term or long term. A short-term memory stores the record of the most recent move history and is used to prevent the search cycling in those recently visited solutions. A long-term memory is based on the record of the whole move history. It is used to guide the search of neighbours of elite solutions (named as intensification search strategy), or not explored, but promising regions (named as diversification search strategy). A simple tabu search only employs the short-term memory. An advanced tabu search uses both short-term and long-term memory. We aim to implement both a simple and an advanced tabu search and take the maximum advantage of the available domain knowledge on graphs, such as vertex labels and vertex-degrees. The remainder of this section gives a more detailed look at the components of our tabu search algorithm.

3.1 Initialisation

Given a target graph G_t and a source graph G_s, we only consider to establish an initial one-to-one vertex correspondences between G_t and G_s, as arc correspondences are dependent on the vertex correspondences. In each iteration, we consider a vertex of G_t, and choose a vertex from G_s from those vertices that are not yet mapped by the following greedy selection rules.

- A vertex $u \in V_t$, and a vertex $v \in V_s$, will be given the highest priority to establish a vertex map if $\alpha_t(u) = \alpha_s(v)$, and the difference between two vertex-degrees is the smallest among the remaining possible vertex maps.
- The second mapping priority will be given to two vertices, u and v, if $\alpha_{t(u)} \neq \alpha_{s(v)}$ for all $v \in V_s$, but the difference between two vertex-degree numbers is the smallest among all remaining possible vertex maps.
- Vertices without any priorities prescribed above will be considered last and assigned by randomly.

3.2 Move and Neighbourhood

In our tabu search algorithm, we define two possible move operators. A move exchanges two vertices map pairs. For example, suppose that $a, b \in V_t$ and $c, d \in V_s$, $f(a) = c$ and $f(b) = d$. After the swap we get a new correspondence f' in which $f'(a) = d$ and $f'(b) = c$. This move is denoted as *move* (d, c). A move can also swap a mapped vertex with a not mapped one according to f. The neighbourhood of f is denoted as $N^*(f)$.

The cardinal number of the complete neighbourhood of a given graph correspondence f is $|V_t| \cdot |V_s|$. The value of $DS_{f'}(G_t, G_s)$ can be calculated by re-evaluating those vertex and arc similarity degrees modified by f'. A move value is the difference between $DS_f(G_t, G_s)$ and $DS_{f'}(G_t, G_s)$, and is denoted by $DS_{move}(f, f')$.

3.3 Tabu Memories

We implement both short-term memory and long-term memory. The short-term memory is a recency-based memory. During the search, once a *move* (i, j) is applied, its reverse *move* (j, i) is labelled tabu-active in the tabu list and will be forbidden to apply for a number of iterations (tabu tenure). A dynamic neighbourhood is maintained by varying the tabu tenure. The purpose of having a varying tabu list length is to give the search the ability to circumvent local minima with different widths [27]. A tenure, denoted by *tenure* (i, j) for *move* (i, j) is:

$$tenure(i, j) = b + e^{(1 + d.DSmove(f, f'))} \tag{6}$$

where b and d are adjustable parameters that can be changed during the search. A big value of b or d corresponds to a desire to increase the tenure range. The moves that produce better results will be tabu-active longer than those that lead to worse results.

The long-term memory is a frequency-based memory that records the number for each move in the search history.

In addition to the above two memories, we maintain another memory that is based on the context of labelled directed graph. Certain moves are set as permanent tabu-active by this memory. According to the similarity measure described in Section 2, the optimal correspondence from G_t to G_s can be only obtained when most vertices with similar vertex-

degrees and the same vertex labels are mapped. By using this domain-knowledge, we forbid any moves that introduce a vertex-pair map if the difference of the two vertex-degrees is greater than a threshold and their vertex labels are different. We set the threshold to the average vertex degree of G_t.

3.4 Search Process

Our tabu search algorithm includes a simple tabu search and an advanced tabu search with the intensification and the diversification. The initial solution is built by the greedy selection rules described in Section 3.1.

The simple tabu search ranks the neighbours of the present solution for choosing the next move. Then it applies the move with the highest move value, if the move is not prohibited by either the recency-based memory or the permanent tabu list. The simple tabu search could be used either as an independent approach or as the sub-unit in the advanced tabu search. In general, the solution produced by the simple tabu search is worse than the solution produced by the advanced tabu search in terms of the value of $DS(G_t, G_s)$.

The advanced tabu search is based on the simple tabu search and it is implemented by employing the search strategies of intensification and diversification alternatively. It commences from the result obtained from the simple tabu search. The intensification starts from the best solution found so far, then clears the recency-based memory while keeping the permanent tabu list, and repeats a simple tabu search again. The diversification also starts from the best solution found so far and clears recency-based memory, keeps the permanent tabu list, and repeats a simple tabu search. However its move value formulation is modified by using the frequency-based memory. In our implementation, the modified move value is denoted as $DS^*_{move}(f, f')$,

$$DS^*_{move}(f, f') = DS_{f'}(G_t, G_s) - DS_f(G_t, G_s) + \ln(1 + Num\_move(i, j)) \qquad (7)$$

where $Num\_move(i, j)$ denotes the total number of moves which involve either vertex i or j, which is recorded in the frequency-based memory.

In this study, we terminate a phase when a solution's similarity degree has not been improved for 100 moves. The best solution found in the whole search process will be returned as the final result.

4. Two Stage Case Retrieval

Real world CBR applications require the process of case retrieval to return the most similar cases in the shortest possible time. However, this goal is very hard to implement for the graph structured CBR systems, not only because of the computational complexity of the problem of case matching, but also because of the difficulties of developing valid retrieval algorithms (indexing schemes) to manage the organisation of the case base. In this section, we propose a two-stage strategy for the process of case retrieval, where a simple tabu search works as the search function and an advanced tabu search is employed for the case matching.

When solving a new problem, the simple tabu search is firstly used to rank all the cases in the case base by calculating the similarity degree approximately. Then a subset of possible similar cases will be presented to the advanced tabu search. Cases in the subset will be evaluated more precisely by the advanced tabu search. Finally those most similar cases will be ranked by similarity degree. An important advantage of the above framework

is that the advanced tabu search could benefit from the experiences obtained from the simple tabu search, so the case retrieval time will be shorten significantly. This process is shown in Figure 1.

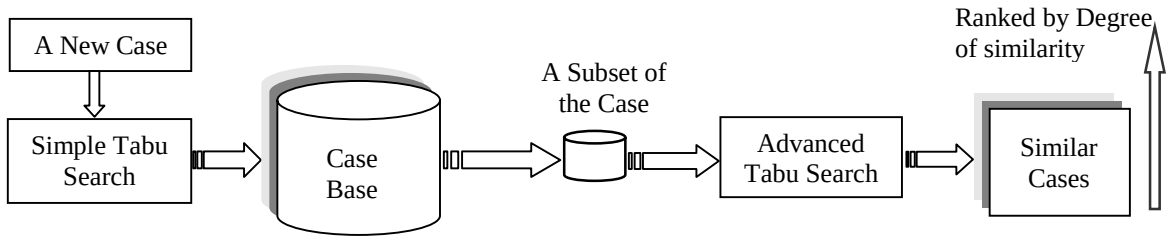

Figure 1. A two stage strategy for the process of graph-structured case retrieval

5. Experiments

The following preliminary results show the performance of a simple tabu search. Our aim is to investigate whether the simple tabu search could find an approximate solution quickly. As far as the authors are aware, benchmark problem instances with known optimal solutions are not available in the literature. To obtain the optimal correspondence between the two graphs is impossible due to the NP-hard characteristic of the problem, so a stochastic simulation is employed to investigate the effectiveness of our approach [18]. Our tabu search approach is implemented in C++ and experiments were conducted on an Athlon 900MHz with 128MB RAM running under Windows version 2000.

A test problem consists of two directed-labelled graphs, a target graph G_t and a source graph G_s. Firstly, G_t is generated randomly. The second graph G_s is created through several steps. In the first step, G_s is a copy of G_t. At this moment, the similarity degree between two graphs (denoted by DS_{record}) is equal to 1, which is obtained by simply mapping the same vertices between the two graphs. This vertex-to-vertex correspondence is called the ideal-map. Secondly, G_s is gradually revised by a series of random revise actions, such as changing labels, adding and deleting vertices and arcs. As result, the ideal-map and DS_{record} will be corrupted in the revision process. DS_{record} can take a value between 0 and 1, where 1 represents that G_t is exactly isomorphic to G_s. DS_{record} will be recalculated after each revise action. Once both graphs are created, they are inputted to the tabu search. We use DS_{tabu} to denote the similarity degree between G_t and G_s obtained by our tabu search approach. The difference between the DS_{record} and DS_{tabu} (denoted by DS_{gap}) is used as the performance evaluation. Because the map associated with DS_{record} is not necessarily optimal solution of the test problem, DS_{gap} can be negative when the tabu search approach finds a solution better than the map. By using this simulation method, we can create different test problems with arbitrary values of DS_{record}.

We initially test the performance of the simple tabu search on small graphs (number of vertices varying from 5 to 50). Next we compared the result with that of the genetic algorithm in [18], as our similarity measure and the stochastic simulation method for our experiments are as same as those are in [18]. We use DS_{ga} to denote the similarity degree found by the genetic algorithm. The test set includes one thousand test problems that are randomly produced and divided into 10 groups with different values for DS_{record}. Table 1 shows the performance comparison of the genetic algorithm and our simple tabu search. The right part of Table 1 shows the results in terms of DS_{gap} obtained by the simple tabu search; the left part shows results obtained by the genetic algorithm. Each row represents one test group that is formed with a corresponding value of DS_{record}. Each number in a field of the table represents the number of test problems of a certain group whose DS_{gap} value

belongs to the range defined in the column title. The bottom row shows the total number of such test problems.

From Table 1, a better performance of the simple tabu search over genetic algorithm can be observed. The total number of the test problems where the DS_{tabu} is greater than DS_{record} is 795, whereas that of the genetic algorithm is 372. The total number of the test problems where the DS_{tabu} is smaller than DS_{record} is 3, whereas with the genetic algorithm is 11. Even more the tabu search is fast enough, as the average processing time is only 372 milliseconds.

Table 1 The simulation results of the simple tabu search compared with the genetic algorithm

Range of DS_{record}	The number of pairs of graphs							
	$DS_{gap} = DS_{record} - DS_{ga}$				$DS_{gap} = DS_{record} - DS_{tabu}$			
	(−0.1, 0)	[0, 0]	(0, 0.1)	[0.1, 0 .4)	(−0.1, 0)	[0, 0]	(0, 0.1)	[0.1, 0 .4)
[0, 0.1)	1	0	1	0	2	0	0	0
[0.1, 0.2)	6	4	2	0	12	0	0	0
[0.2, 0.3)	5	6	3	1	15	0	0	0
[0.3, 0.4)	38	21	0	2	61	0	0	0
[0.4, 0.5)	42	28	0	1	68	1	2	0
[0.5, 0.6)	54	40	0	1	95	0	0	0
[0.6, 0.7)	88	83	0	0	169	1	1	0
[0.7, 0.8)	76	117	0	0	188	5	0	0
[0.8, 0.9)	55	172	0	0	166	61	0	0
[0.9, 1.0)	7	146	0	0	19	134	0	0
[0, 1.0)	372	617	6	5	795	202	3	0

Our next experiments are designed to evaluate the performance of the simple tabu search on large graphs. We first generated 360 test problems and divided them into three test problem groups of the same size. The average number of vertices in a graph A, B and C is 100, 300 and 500 respectively. Each group is divided into six subgroups of equal sizes by the ranges of DS_{record} varying from 1.0 to 0.4.

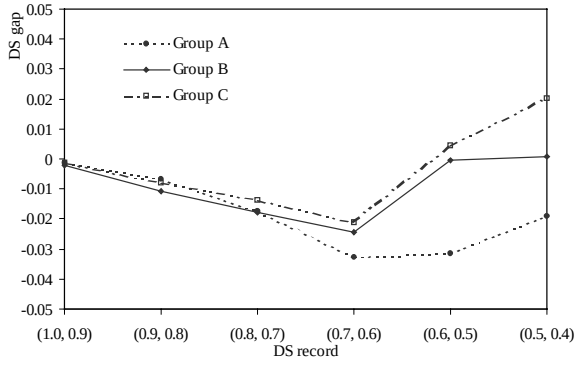

Figure 2. Performance comparison of the test groups with graph of different sizes.

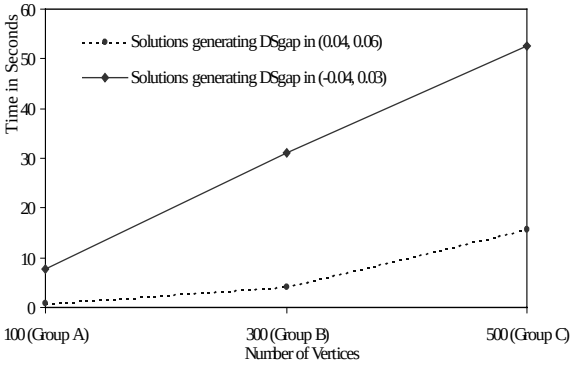

Figure 3. Time needed for generating solutions with different qualities.

Figure 2 shows the results of the three test groups in terms of DS_{gap}. In Figure 2, six points on the x-axis are used to represent six sub-groups in a test group. It represents the average value of DS_{gap} for the 20 problems in its sub-group. The performance of the simple tabu search for the test problems of the group B and C in the range of DS_{record} from 0.4 to 0.6 is slightly worse than for others.

Figure 3 compares the time in seconds needed for generating solutions of different qualities. In this figure, three points on the x-axis are used to represent three test groups. A point on the solid curve represents the average running time for generating a solution for one of the 120 test problems in its corresponding group, which renders DS_{gap} ranging from -0.04 to 0.03. A point on the dotted curve represents the average running time for

generating a slightly worse solution for one of the 120 test problems in its corresponding group, which renders DS_{gap} ranging from 0.04 to 0.06. The second experiment illustrates that the simple tabu search approach developed so far is capable of generating solutions of high quality for large graphs speedily, which render the similarity degree values better than or close to the values of DS_{record}.

6. Conclusion

In this paper we present a tabu search approach for the graph-structured case retrieval problem. Our preliminary experimental results show the efficiency of the simple tabu search for the first step in case retrieval. The advanced tabu search for the case matching will be implemented in our future works. We believe that the proposed two-stage strategy for retrieving graph-structured cases is valuable for a number of real-world CBR applications where the graph has several hundred vertices and there are several hundred cases in the case base. Future work will also apply our tabu search approach in a CBR system designed to solve university-timetabling problems [8].

Acknowledgements

We would like to thank the anonymous reviewers for their careful work in evaluating an earlier version of this paper.

References

[1] Kolodner, J., Case-Based Reasoning, Morgan Kaufmann Publishers, (1993).
[2] Watson, I., 'Case-based reasoning is a methodology not a technology', Knowledge-Based Systems, 12, 303-308, (1999).
[3] Riesbeck, C., and Schank, R., Inside Case-based Reasoning, Lawrence Erlbaum, (1989).
[4] Aamodt, A., and Plaza, E., 'Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches', AI Communications, 7-1, 39-59, (1994).
[5] Mántaras, R.L., and Plaza, E., 'Case-Based Reasoning: An Overview', AI Communications, 10, 21-29, (1997).
[6] Sanders, K.E., Kettler, B.P., and Hendler, J.A., 'The Case for Graph-Structured Representations', ICCBR 97, 245-254, Berlin: Springer-Verlag, (1997).
[7] Watson, I., and Marir, F., "Case-based reasoning: A review", The Knowledge Engineering Review, 9-4, 327-354, (1994).
[8] Burke, E.K., MacCarthy, B., Petrovic, S., and Qu, R., 'Case-based Reasoning in Course Timetabling: An Attribute Graph Approach', ICCBR-2001, 90-104, Vancouver, Canada, (2001).
[9] Kettler, B.P., Case-based Planning with a High Performance Parallel Memory. Doctoral Dissertation (Technical Report CS-TR-3561), University of Maryland at College Park, (1995).
[10] Sanders, K.E., CHIRON: Planning in an Open-Textured Domain, Technical report 94-38, Computer Science Department, Brown university, (1994).
[11] Fung P.W., and Kemp, R.H., 'On developing case-based tutorial systems with conceptual graphs', Springer-Verlag, Lecture Notes in Artificial Intelligence No. 1640, 214-229, (1999).
[12] Praehofer, H., and Kerschbaummayr, J., 'Development and Application of Case-Based Reasoning Techniques to Support Reusability in a Requirement Engineering and System Design Tool', Engineering applications of Artificial Intelligence, 12, 717-731, (1999).
[13] Coulon C.H., 'Automatic indexing, retrieval and reuse of topologies in architectural layouts'. The Global Design Studio-Proceedings of the 6th International Conference on Computer-Aided Architectural Design Futures, 577-586, Singapore, (1995).
[14] Coello, J.M.A., and Santos, R.C., 'Integrating CBR and Heuristic Search for Learning and Reusing Solutions in Real-Time Task Scheduling', ICCBR 99, 89-103, Germany, (1999).

[15] Garey, M.R., and Johnson, D.S., Computers and Intractability a Guide to the Theory of NP-Completeness. W.H.Freeman, (1977).
[16] Bunke, H., and Messmer, B.T., 'Similarity Measures for Structured Representations', First European Workshop on Case-Based Reasoning, (1993).
[17] Messmer, B.T., and Bunke, H., 'Efficient Subgraph Isomorphism Detection: A Decomposition Approach'. IEEE transactions on knowledge and data engineering, 12-2, 307-323, (2000).
[18] Wang, Y., and Ishii, N., 'A method of similarity metrics for structured representations', Expert Systems with Applications, 12-1, 89-100, (1997).
[19] Corneil, D.G., and Gotlieb, C.C., 'An efficient algorithm for graph isomorphism', Journal of the Association for Computing Machinery, 17, 51-64, (1970).
[20] Ullman, J.R., 'An algorithm for subgraph isomorphism', Journal of the Association for Computing Machinery, 23, 31-42, (1976).
[21] Haralick, R.M., and Elliott, G., 'Increasing tree search efficiency for constraint satisfaction problems', Artificial Intelligence, 14, 263-313, (1980).
[22] Williams, M.L., Wilson, R.C., and Hancock, E.R., 'Deterministic search for relational graph matching', Pattern Recognition, 32-7, 1255-1271, (1999).
[23] Herault, L., Horaud, R., Veillon, F., and Niez, J.J., 'Symbolic image matching by simulated annealing', Proc. British Machine Vision Conf., 319-324, (1990).
[24] Cross, A.D.J., Wilson, R.C., and Hancock, E.R., 'Inexact graph matching with genetic search', Pattern Recognition, 30-6, 953-970, (1997).
[25] Cross, A.D.J., Myers, R., and Hancock, E.R., 'Convergence of a hill-climbing genetic algorithm for graph matching', Pattern Recognition, 33, 1863-1880, (2000).
[26] Glover, F., and Laguna, M., Tabu Search, Kluwer Academic Publishers, 1997.
[27] Ryan, J., "The depth and width of local minima in discrete solution spaces", Discrete Applied Mathematics, 56-1, 75-82, (1995).