Soft Comput (2007) 11: 391–395
DOI 10.1007/s00500-006-0094-2

FOCUS

Yan Li · Xi-Zhao Wang · Ming-Hu Ha

An on-line multi-CBR agent dispatching algorithm

Published online: 4 April 2006

Abstract Case-based reasoning (CBR) is an effective and fast problem-solving methodology, which solves new problems by remembering and adaptation of past cases. With the increasing requests for useful references for all kinds of problems and from different locations, keeping a single CBR system seems to be outdated and not practical. Multi-CBR agents located in different places are of great support to fast meet these requests. In this paper, the architecture of a multi-CBR agent system is proposed, where each CBR agent locates at different places, and is assumed to have the same ability to deal with new problem independently. When requests in a request queue are coming one by one from different places, we propose a new policy of agent dispatching to satisfy the request queue. Throughout the paper, we assume that the system must solve the coming request by considering only past requests. In this context, the performance of traditional greedy algorithms is not satisfactory. We apply a new but simple approach – competitive algorithm for on-line problem (called ODAL) to determine the dispatching policy to keep comparative low cost. The corresponding on-line dispatching algorithm is proposed and the competitive ratio is given. Based on the competitive algorithm, the dispatching of multi-CBR agents is optimized.

Keywords CBR · Multi-CBR agent · On-line problem · Competitive algorithm

1 Introduction

With the widespread popularity of the Internet and World Wide Web, there are hundreds or thousands of specific-topic information sources distributed in different locations. Moreover, a wide variety of requests for useful references to specific problems are increasing from different places. Storing and dealing with all the data in a single case-based reasoning (CBR) system seems to be outdated and not practical and efficient. Some research work shows that the multi-agent system is an effective and promising strategy to improve the performance of the problem-solving ability [1,2]. Multi-agent systems offer a new paradigm where learning techniques can be useful. Here, each agent learns individually and also learns when to cooperate in order to improve its performance.

In this paper, an agent refers to a CBR system. CBR gives the multi-agent systems paradigm the capability of autonomously learning from experience. All the discussions are in the context of CBR systems. For fast meeting requests from different places, the architecture of a multi-CBR agent system is proposed. Each agent locates in different place from others and has independent ability to solve all incoming requests. When a request arises from a certain place, dispatching different agents will have a different cost (such as the distance to move). In the proposed system architecture, attention is paid to satisfying a request as soon as possible and minimizing system-handling cost, while not knowing where the future requests will occur. In this context, the dispatching policy of selecting the shortest distance agent is not always a good solution. To address such a multi-CBR agent-dispatching problem, a new approach called competitive algorithm is applied, through which we can control the cost in a certain scope (refers to the competitive ratio) of the optimal solution whatever the request queue is.

The remainder of this paper is organized as follows. In Sect. 2, the on-line agent-dispatching problem (OLAP) is described and the traditional greedy algorithm is shown not to be optimal. Sect. 3 proposes the architecture of the multi-CBR agent system and four concerning assumptions are also given. The basic concepts of on-line problem and competitive algorithm are reviewed, then a kind of competitive algorithm called on-line multi-CBR agent-dispatching algorithm is developed. The comparison results between greedy algorithm and the proposed algorithm is given through experimental results. Finally, the conclusion is presented in Sect. 4.

Y. Li (✉) · X.-Z. Wang · M.-H. Ha
College of Mathematics and Computer Hebei University,
Baoding, Hebei, 071002, China
E-mail: csyli@comp.polyu.edu.hk
E-mail: wangxz@mail.hbu.edu.cn
E-mail: mhha@mail.hbu.edu.cn

2 On-line multi-CBR agent dispatching problem

In this paper, in the context of CBR system, we focus on the agent-dispatching problem. When we consider a system including multi-CBR reasoners (which can be considered as a multi-CBR agent system), the problem of how to dispatch the reasoners to meet the requests from different places should not be ignored. When a request queue is coming one by one, which agent (CBR reasoner) should be dispatched is an important issue in a multi-CBR agent system to keep the system cost as low as possible. So far, there has been little research to address such a topic. In this paper, to achieve flexibility and efficiency, in the proposed multi-CBR agent system, we assume that each CBR agent locates at different places, while has the same ability to solve requests (new cases, or problems) independently. When a request occurs in a certain place, the system must dispatch one of the agents to meet the request. The cost incurred to serve the request can be regarded as the distance the agent moved. Then the cost of a queue of requests is the sum of the distances each agent moved.

We consider such a situation which is another important assumption here: the system must meet a request by considering only past requests. That is, a request in the request queue may come one by one; the system must make a decision of dispatching which CBR agent before the future requests coming. The choice of the agent to meet a request at each step must be made on-line without knowledge of future requests. After the request is met, the corresponding agent then stays at the place the request occurred. This kind of problem is called on-line multi-CBR agent -dispatching problem.

Greedy algorithm is an often-used method in this situation. Its main idea is always selecting the agent which moves the shortest distance to meet the coming request. In this paper, we argue that greedy algorithms do not perform well for the mentioned on-line multi-CBR agent- dispatching problem. A simple example is presented as follows to illustrate the reason why greedy algorithm is far from optimal.

2.1 An example

Give a line on which there are three points. Initially, two CBR agents, Ag_1 and Ag_2, are located on the line, one at point L_2 and another at point L_3, as shown in Fig. 1. The distance from L_1 to L_2 is one unit, and from L_2 to L_3 is two units. The cost of moving the agents is proportional to the distance. Here, we assume when a new request (new case or problem) occurs the two agents are both in their ready states to move.

If we know the request queue R in advance, then the problem of dispatching is off-line and dynamic programming may be applied to solve it easily. In this paper, we assume that requests in the queue occur sequentially.

Therefore, whenever the system makes a decision of which agent to dispatch, the system only knows part of the request queue which has occurred. Thus, the problem belongs to an on-line problem (see the definition of on-line problem in Sect. 3).

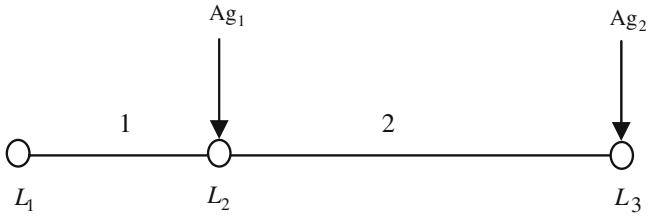

Fig. 1 On-line two CBR agent system dispatching

Greedy algorithm is applied to illustrate why it performs badly under certain conditions. Consider such a request queue $\underbrace{L_1L_2L_1L_2\cdots L_1L_2}_{m}$. There are total m requests where the request occurs from places L_1 and L_2 repeatedly. With a greedy algorithm, whose main idea is always choosing the nearest agent to meet the request, a CBR agent Ag_1 will travel between the point L_1 and L_2 on m occasions. The value of total cost can be represented as m. If we know the request queue beforehand, that is, if the problem is an off-line problem, the optimal solution (when m is greater than 3) can easily been obtained by moving Ag_1 to L_1 and then moving Ag_2 from the point L_3 to L_2. The following requests are always met without moving any agent more after that. Obviously, the cost of this optimal dispatching is only 3. We can see that the ratio of the values of the greedy algorithm to the optimal dispatching is $(m-1)/3$, which tends to be infinite big when m becomes infinite big. The greedy algorithm performs badly because its solution may be very far from the optimal solution for this problem. The reason is this greedy algorithm let one agent very busy while the other is idle.

In this context, a kind of competitive algorithm is proposed to solve this on-line multi-CBR agents dispatching problem, which refers to on-line multi-CBR agent dispatching algorithm (ODAL) in this paper.

3 On-line multi-CBR agent-dispatching algorithm

3.1 Basic concepts

Before the ODAL is proposed, some related concepts are given as follows.

Definition 1 *On-line problem refers to a problem that needs the solver to make decisions with partial information.*

Definition 2 *Competitive algorithm is a kind of algorithm to solve on-line problem.*

The main idea of competitive algorithm is: it does not consider the absolute behavior of the algorithm, instead, it considers the ratio between the algorithm's behavior and the optimal behavior for a given problem instance; that is, with respect to maximizing benefit or minimizing cost problems, competitive algorithms may keep the solution to be in a certain scope of the optimal solution.

Definition 3 *The ratio between the solution of a competitive algorithm and the optimal solution is called competitive ratio.*

The underlying application of competitive ratio for cost problems is the ratio between the solution of competitive algorithm A and the solution which achieves the minimum cost. While the competitive ratio for benefit problems is the ratio between the solution of algorithm A and the solution which achieves the maximum benefit. Thus, the smaller the competitive ratio is, the more similar the solution of algorithm A to the optimal solution and the better the algorithm A is.

Though on-line problems and competitive algorithms are studied for about only 30 years, they have found a wide applications [3–5]. Because of the incomplete information in on-line problems, seeking a competitive algorithm which has an acceptable competitive ratio is a safe and reasonable way.

3.2 On-line dispatching algorithm for two CBR agents

Based on the basic concepts given in Sect. 3.1, a competitive algorithm is given to solve the on-line dispatching problem for two CBR agents.

For the on-line dispatching problem illustrated in Fig. 1, we assume that $\mathrm{Ag}_i (i = 1, 2)$ has moved D_i units, the shortest distance from point i to j is d_{ij}, and the agent request occurs at the point $k(k = 1, 2, 3)$. We give a competitive algorithm called on-line dispatching algorithm for two CBR agent system as follows:

(a) if $D_i + d_{ki} < D_j + d_{kj}$, then move Ag_i to point k to satisfy the new request;
(b) if $D_i + d_{ki} > D_j + d_{kj}$, then move Ag_j to point k to meet the new request;
(c) if $D_i + d_{ki} = D_j + d_{kj}$, then select one of the agents (1 or 2) at random.

Using this kind of strategy, consider the request queue $\underbrace{L_1 L_2 L_1 L_2 \cdots L_1 L_2}_{m}$. It is easy to see the cost value of the solution is 5. In fact, for any request queue in Fig. 1, it can be shown that according to this competitive algorithm, the total cost both the two agents move will be less than or equal to double the cost of the optimal solution, say, the competitive ratio is 2, which is a comparative small number.

Comparing to the greedy algorithm, the performance of our proposed competitive algorithm is much better. As discussed in Sect. 2.1, the ratio between cost of the greedy algorithm to the optimal algorithm is $(m - 1)/3$, when m is big, the ratio will intend to be big at the same time, which leads to a poor result.

From the previous discussion, we can say that the on-line multi-CBR agent-dispatching algorithm (ODAL) is a very safe strategy when the future information is not complete. The poor performance of the greedy algorithm can be effectively avoided by this kind of methods.

In the following sections, we discuss a more complex problem of on-line multi-CBR agent dispatching, in which there are k CBR agents and each of them occupies a place. Each request in a request queue may occur at $(k+1)$ different locations. The corresponding ODAL is also given.

3.3 Structure of multi-CBR agent system

To achieve flexibility and efficiency, in this section, a multi-CBR agent system including more agents is introduced. At the same time, requests may occur at more different places.

We assume there are k $(k > 2)$ agents, $\mathrm{Ag}_1, \mathrm{Ag}_2, \ldots, \mathrm{Ag}_k$, in the proposed multi-CBR agent system, and each of them $\mathrm{Ag}_i (i = 1, 2, \ldots, k)$has a different locate $L_i (i = 1, 2, \ldots, k)$. There is a request queue $R = (r_1, r_2, \ldots, r_m)$which includes m request, and each request r_i may occur on $(k + 1)$ different locations, $L_i (i = 1, 2, \ldots, k + 1)$. Here, the first k locations are the same as that of the k agents. Two simple structures are illustrated in Figs. 2 and 3. Certainly, there are more complex structures in different contexts.

3.4 Assumptions

In our multi-CBR agent system, there are several important assumptions:

1. Each agent of the multi-CBR agent system including k agents has the same ability to solve any request (new case, or problem), that is, each agent is an independent unit.

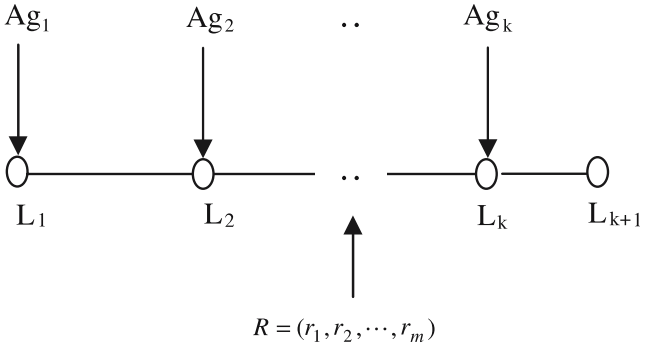

Fig. 2 One possible structure of on-line multi-CBR agent system

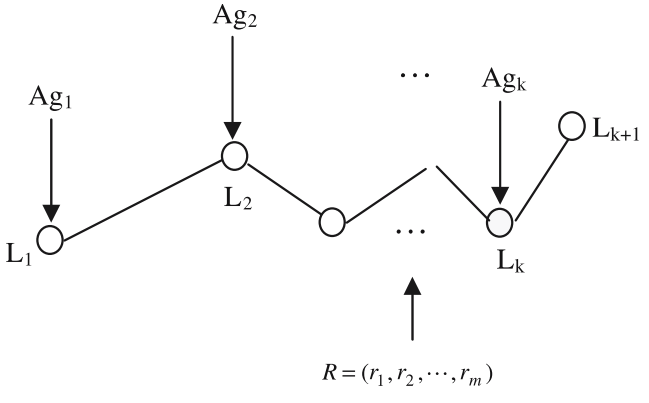

Fig. 3 Another possible structure of on-line multi-CBR agent system

2. To achieve efficiency and flexibility, different CBR agents in the multi-CBR agent system locate at different places.
3. Reasonably, requests may come from different locates. Here, the requests occur from $(k+1)$ different places, which include the k points of agent location. When each request in a request queue is met, the corresponding agent moves to the location of the request.
4. The last and the most important assumption is, when a request in a request queue occurs at a certain place, the system has to decide which agent should be dispatched to meet the request with respect to minimize the cost without knowledge about future requests. Essentially, it is an optimization problem with incomplete information, that is, an on-line problem.

In assumption (4), the cost is considered to be proportional to the distances of the k agents moved. In the following section, competitive algorithms are proposed to solve the on-line dispatching.

3.5 ODAL for proposed multi-CBR agent system

In this section, with respect to the multi-CBR agent system developed in previous subsections, we propose a kind of on-line dispatching algorithm.

The given ODAL is directly derived from the algorithm proposed in the simple example illustrated in Fig. 1. Here, we assume that when a request (a new case) r_i in a request queue R occurs on a certain location, the ith CBR agent Ag_i(locating the ith location L_i) has moved a distance D_i to solve previous requests. Then the on-line dispatching policy is ODAL.

ODAL For each CBR agent $\mathrm{Ag}_i(i = 1, 2, \cdots, k)$ and the request r_i,

(1) Compute $D_i + d(\mathrm{Ag}_i, r_i)$, where $d(\mathrm{Ag}_i, r_i)$ means the distance from $\mathrm{Ag}_i$to $r_i(i = 1, 2, \ldots, \mathrm{k})$.
(2) Select the agent which minimize $D_i + d(\mathrm{Ag}_i, r_i)$ to solve r_i.

The proposed algorithm is very simple, but it is very effective. A good competitive ratio can be achieved through the algorithm.

The competitive ratio The competitive ratio of ODAL is k [6] in our proposed structure of multi-CBR agent system, under the given assumptions.

The proof is omitted here.

4 Experimental results

Comparative study between the greedy algorithm and the competitive algorithm is implemented. In this section, a simple experiment is developed to show the feasibility and effectiveness of our proposed ODAL.

The experiment is set up as follows.

First, let $k = 3$, say, there are three CBR agents Ag_1, Ag_2, Ag_3in the multi-CBR agent system in our proposed structure. The three agents locate three different places, abstractly, three points L_1, L_2, L_3 in the structure graph.

Second, let the new case (problem) $r_i(i = 1, 2, \ldots, 6)$in a request queue R occur on four different places L_1, L_2, L_3, L_4 at random. When making decision of dispatching which agent to solve the occurred request, the place where future requests come from is not informed.

Third, let the cost of solving the request queue R is the summation of distances each agent traveled. The distance for $\mathrm{Ag}_k(k = 1, 2, 3)$ solving $r_i(i = 1, 2, \ldots, 6)$is represented by d_{ki}, which is defined as:

$$d_{ki} = \begin{cases} 0 & k = i \\ |k - r_i| & k \neq i \end{cases}$$

where $r_i = i$, and $k = 1, 2, 3$, $i = 1, 2, \ldots, 6$.

$r_i = i$ means that when a request occur at random in L_i, we use the index i to denote it.

Then greedy algorithm and our new approach are tested, respectively. In the greedy algorithm, we always select the agent which moves the shortest distance to solve a certain problem, that is, for each r_i, choose the agent who will move the distance of $\min\{d_{1i}, d_{2i}, d_{3i}\}$. While in our proposed ODAL algorithm, we first compute the overall distances $d_k = \sum_{i=1}^{m} d_{ki}$ of each agent $\mathrm{Ag}_k(k = 1, 2, 3)$ traveled when a problem $r_m(m \geqslant 6)$ coming, and then select the agent having the shortest overall distances $\min\{d_1, d_2, d_3\}$. Finally, the cost of the system is computed as the summation distance of all the agents moved, which denoted by D_g and D_o for greedy algorithm and on-line dispatching algorithm, respectively.

The request queue is generated at random by different times from 10 to 100, then compute the average cost of each algorithm represented by $\overline{D}_\mathrm{g}$ and $\overline{D}_\mathrm{o}$. The results are illustrated in the Fig. 4. Obviously, the average cost of our proposed ODAL algorithm is much lower than that of greedy algorithm.

5 Conclusion

With the increasing requests for useful information service, keeping a multi-CBR agent system is an effective way.

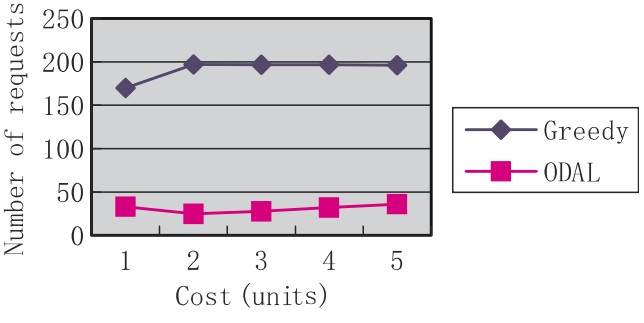

Fig. 4 Comparison of cost

Consider different requests come from different locations; and the system must response immediately to determine which agent to dispatch to solve the request with respect to the minimum cost. When a request occurs at a certain place, the system must make a decision without knowing the future requests. In this context, the performance of greedy algorithms are shown to be unsatisfied under certain request conditions. While another safety method called competitive algorithm is introduced to address this optimization problem. The corresponding competitive algorithm for our proposed multi-CBR agent system has been given, and the competitive ratio is equal to the number of agents which keeps the solution of dispatching is in a certain scope of optimal solution. The experiment result shows that our OLAP dispatching strategy has significantly reduced the cost of agents.

Acknowledgements This research is supported by the National Natural Science Foundation of China (60473045/60573069); the Natural Science Foundation of Hebei Province (603137).

References

1. Plaza E, Ontañòn S (2001) Ensemble case-based reasoning: collaboration policies for multiagent cooperative CBR. In: Proceedings of ICCBR-01, pp 437–451
2. Ontañon S, Plaza E (2001) Learning when to collaborate among learning agents. In: Proceedings of machine learning: EMCL-01, pp 394–405
3. Papadimitriou CH, Yannakakis M (1991) Shortest paths without a map. Theor Comput Sci 84:127–150
4. Chrobak M, larmore L (1992) The server problem and on-line games. DIMACS Ser Discrete Math Theor Comput Sci 7:11–64
5. Chekuri C, Motwani R, Natarajan B, Stein C (2001) Approximation techniques for average completion time scheduling. SIAM J Comput 31:146–166
6. Manasse MS, McGeoch LA, Sleator DD (1990) Competitive algorithms for server problems. J Algorithms 11:208–230