An Evolutionary Algorithm for Adaptive Online Services in Dynamic Environment

Alfredo Milani[1,2], Clement Ho Cheung Leung[1],
Marco Baioletti[2], and Silvia Suriani[2]

[1] Hong Kong Baptist University, Department of Computer Science , Hong Kong
milani@unipg.it
[2] Universitá degli Studi di Perugia, Dipartimento di Matematica e Informatica,
Via Vanvitelli 1, I-06100 Perugia, Italy

Abstract. An evolutionary adaptive algorithm for solving a class of online service provider problems in a dynamical web environment is introduced. In the online service provider scenario, a system continuously generates digital products and service instances by assembling components (e.g. headlines of online newspapers, search engine query results, advertising lists) to fulfill the requirements of a market of anonymous customers. The evaluation of a service instance can only be known by the feedback obtained after delivering it to the customer over the internet or through telephone networks. In dynamic domains available components and customer/agents preferences are changing over the time. The proposed algorithm employs typical genetic operators in order to optimize the service delivered and to adapt it to the environment feedback and evolution. Differently from classical genetic algorithms the goal of such systems is to maximize the average fitness instead of determining the single best optimal service/product. Experimental results for different classes of services, online newspapers and search engines, confirm the adaptive behavior of the proposed technique.

1 Introduction

Mass services based on new information technologies represent a challenging domain for providing adaptive services to a population of anonymous customers[1]. Purely digital service/products are generated and delivered instantaneously on demand to thousands of users. For instance, the headlines of an online newspaper, web banners advertising special offers, voice menu systems proposing valuable offers to mobile phone customers, list of documents returned after a search engine query, are all example of products which are submitted to a mass of individual users, which in a short time decide to browse, to buy, to accept or to neglect the proposed product. It is worth noticing the dynamical nature of these purely digital products: the items to compose (e.g. news, available offers,indexed documents) and the customer preferences rapidly change over time. It is then required a quick adaptation of the products to the changing conditions of the market environment.

M. Giacobini et al. (Eds.): EvoWorkshops 2008, LNCS 4974, pp. 626–632, 2008.

User modeling approaches [2] are often hard to apply to this scenario for a number of reasons: short duration and anonymity of internet connection, privacy reasons, and customers changing preferences do not allow to build a significant user model. Moreover a model of the many services or products cannot be built since they are short lived and disappear very quickly. Some works have introduced "real world" issues into the genetic algorithm (GA) loop. For instance [8] proposes an interactive approach where the user provides the fitness function, [3,4] also interacts with the external world in order to optimize a behavior. Most approaches to adaptation based on GA adopt a classical machine learning "offline" approach [5,6,7,9,10]): in a first "training phase", they evolve the solutions in a "virtual training environment" eventually returning the best one. The offline method fails when the domain of services is dynamically evolving over time and the customer responses can vary as well.

The evolutionary schema we propose is both interactive and adaptive [11], it extends the GA framework with fitness from the external environment, exploiting the optimization [4] and adaptive features of GA operators.

2 The Online Service Provider Problem

Let C a domain of components, S the set of services which can be assembled with elements from C,R a compositional structure $R : C^m \rightarrow S$ where $s = R(c_1, ..., c_m)$ is a service assembled by instantiating the compositional structure R with m components $c_i \in C$, let $f : N \times S \rightarrow [0, 1]$ a *dynamic fitness* function, defined over nonnegative integer N and services, where $f(t, s)$ represents the customers satisfaction for service s at time t, let n the number of service requests per time instant and S_t the set of services delivered at time t where $|S_t| = m$. The Online Service Provider(OSP) problem consists in devising a production strategy which maximizes the overall fitness over a given interval of time $[0, t_{max}]$

$$f_{tot} = \sum_{i=0}^{t_{max}} \sum_{s \in S_i} f(i, s)$$

where the function f is unknown to the systems, i.e. the evaluation of service s at time t is made by interacting with the external agents to which the services/products are delivered. We assume that all the external agents give the same product the same evaluation at a given time. Different hypotheses about availability of components, and constraints on the compositional structure apply to different classes of online services. In the following we will assume that the compositional structure is fixed and the components are available in unlimited number of copies, as usually pure digital products are. Moreover the product evaluation is assumed to be component *additive*. The agents evaluate a service s at time t with a dynamic fitness equal to the sum of satisfaction degrees of the components c_i, $i = 1, \ldots, m$ of s.

3 An Algorithm for Online Service Providers

The algorithm we have designed for OSP problem is based on a standard evolutionary scheme driven by selection, crossover and mutation operators. The service provider assembles components and grows up a population of n services, moreover it alternates an evolution phase, in which the service provider generates the next population of n service instances by means of genetic operators, with an interactive fitness evaluation phase. Note that the system can evaluates a service only by the satisfaction degrees expressed by the external agents. In general the system cannot compute the satisfaction degree of a product which is not delivered, or the satisfaction degree of the single components.

3.1 Classes of Online Service Providers

The two classes of OSPs which have been analyzed are *search engine* and *online newspaper* . Despite of the apparent differences their structures are quite similar: they both verify the hypothesis of fixed compositional structure with a fixed size (newspaper have usual a fixed structure and search engines return a fixed number of links per page, they can be easily represented by vectors). In both cases the fitness model is mainly additive: the more relevant the news, the more relevant is the newspaper front page; the more relevant the objects are to query q, the more relevant is the search engine answer list.

Search Engine. Components domain: in the case of the search engine it consists of the set of available documents/objects which can be potentially returned as an answer to a query term q. Compositional structure R is the list of m objects returned as answer. Constraints on R: all the objects links in an answer list must be distinct. Population: is the set of n answer lists returned to the users which have submitted query q during the time instant.Representation: each individual answer list is easily represented by an ordered vector of m entries, where each entry represent a link to an object relevant to the query. The feedback reflects the relevance of the answer list with respect to the user query. The feedback acquisition phase can be realized to monitor the numbers of clicks received by the answer list, the user session duration, user activity patterns, the amount of data exchanged.

Online Newspaper. Components domain: it consists of the set of available news, and for each single news the set of possible candidate titles and/or pictures. The compositional structure R is a set of m news, distributed in the front page, the constraints on R require that: all news in a front page must be distinct; news are grouped in categories (i.e. sports, politics, internal affair etc.); news can appear only in a category to which they belong, news categories can overlap (e.g.the same news can potentially appear under different categories). Population: is a set of n front pages most recently distributed to the customers. Representation: each individual newspaper is represented by an ordered vector of m entries, where each entry contains news is, title/headline and picture. A technique of feedback acquisition similar to the previous one can be used. The feedback reflects the interest of the reader on the delivered front page.

3.2 Representation and Algorithm Structure

An instance of the dynamical OSP problem is characterized by a tuple (C, R, n) where C is the set of components, R is the compositional structure which defines the constraint of the consistent services, n is the amount of service requests per time instant. The algorithm is also parametrized by the classical pair (p_c, p_m),i.e. probabilities of crossover and mutation, note that n also represents the population size.

The representation of components and services is straightforward. The components (i.e. the domain set for genes) are represented by unique identifiers in $\{1, \ldots, N\}$. The services (i.e. query answer lists, or newspaper front pages) are represented by sequences of m component identifiers, with no replications, services correspond to chromosomes of the population.

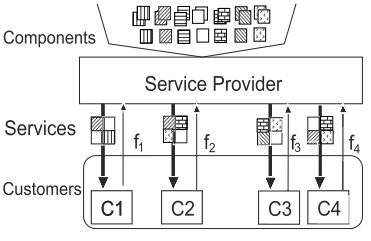

Fig. 1. The Online Service Provider Scenario

The evaluation phase is realized by a procedure, external to the system, which returns for each service s_i in the population at time t, the values of $f(t, s_i)$. These values are used as the fitness of the chromosome.

The evolution phase uses a *selection* procedure to select the chromosomes for the crossover phase, a standard roulette–wheel method is applied, where the fittest individuals are more likely to be selected.

3.3 Crossover and Mutation

The *crossover operator* used in our algorithm is a one–point crossover. The gene sequence of the two chromosome parents are cut in two subsequences of r and $m - r$ elements each, where r is a random number between 1 and $m - 1$. Each of the two chromosome outbreeding is created by taking the union of two subsets coming from different parents. A repairing phase can be necessary after crossover in order to guarantee distinct components in the outbreeding services: if a child has one or more duplicated genes, the replacing components are randonly selected from the parents, or by mutation operators when not possible.

Three different *mutation operators* are used. Explorative mutation operator replaces a component with a randomly chosen component which does not appear in any other chromosome. Full randomized mutation operator replaces a component with a randomly chosen component among all the available components.

Exploitative mutation operator try to use the knowledge acquired in the evaluation phase in order to prefer the components which are likely to give a higher contribution to the chromosome fitness. Since this information is not available to the system, a rough estimate of a quantity proportional to $f_c(t, c_i)$, for each component c_i, is computed by averaging the degree of all the chromosomes in which c_i appears. In the case of online newspaper the mutation operator also takes into account that components are triple and they are organized into topics or categories, i.e. the domain of mutation is the category, and triple elements, like pictures and titles, can also mutate independently.

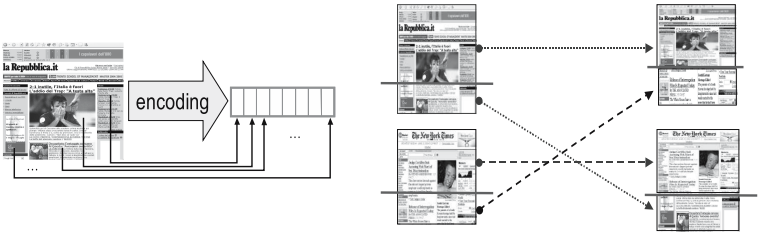

Fig. 2. Component based encoding and single point crossover for online newspaper

4 Experiments

Experiments have focused on the *search engine* and *online newspaper* OSP problems. A special module has been designed to simulate the evaluation given by the external agents and the dynamical changes of the domain by a random uniform distribution of hidden values. The rate of domain evolution over the time is directed by two parameters: probability of change,p_d, and percentage of change, p_a, which have been then varied systematically.

For both classes of experiments have been used the same general parameters: a component domain size of $|C| = 200$ components, each service made of $m = 10$ components (chromosome length), the service provider receives/delivers $n = 20$ service requests per time instant (population size), and optimizes on a time interval $t_{max} = 2000$ (number of generations).

After an initial phase of parameters tuning the probability of crossover and probability of mutation parameters has been fixed to $p_c = 0.15$ and $p_m = 0.09$. The results obtained from the simulations are encouraging and do not significantly differ for both OSP classes. The graph in Figure 3 shows the performances of the system when $p_a = 0.05$. The values of the parameter p_d are shown in correspondence of the line depicting the mean fitness values we obtained. The line for $p_d = 0$ represents the static case, while the line for $p_d = 1$ represents the opposite case when the 5% of fitness is changed at each generation. It is worth to noticing that also in this case the algorithm is still performant. The mean value for the fitness is 0.67 with respect the initial mean value equal to 0.6, with an improvement of more than 10%. An interesting result obtained in the experiments is that the overall fitness obtained by the algorithm appears to be

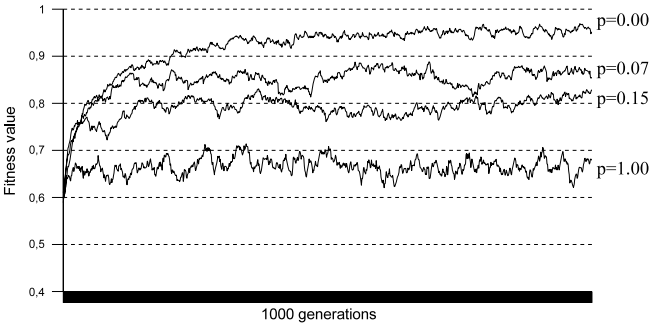

Fig. 3. Results for the online newspaper dynamic OSP problem

invariant with respect to the product $p_d \cdot p_a$, i.e. when the probability of a change in a single component is constant. A possible explanation of these phenomenon is that since p_d is directly related with the time rate of dynamical change, a low value of p_d gives the algorithm more time to adapt to the changes.

5 Conclusions

A general interactive and adaptive evolutionary scheme has been introduced, for two classes of dynamic Online Service Provider problems characterized by component additive fitness: *search engine* and *online newspaper*. The goal of OSP is to maximize the average user fitness while producing and adapting services in a dynamical environment. Experimental results show that the technique converges and adapts to changes for both OSP classes. An interesting properties of the adaptive behavior of the scheme is the ability to recover a given amount of changes despite of their distribution over the time, while preserving the average fitness. Future work will regard different classes of interactive OSP problems where the external fitness is not additive with respect to the components or it is context dependent.

References

1. Binder, J., Koller, D., Russell, S., Kanazawa, K.: Adaptive probabilistic networks with hidden variables. Machine Learning 29(2–3), 213–244 (1997)
2. Kobsa, A., Wahlster, W.: User models in dialog systems. Springer, London (1989)
3. Holland, J.: Adaptation in Natural and Artificial Systems. University of Michigan Press (1975)
4. Whitley, D.: An overview of evolutionary algorithms: practical issues and common pitfalls. Information and Software Technology 43(14), 817–831 (2001)
5. Masui, T.: Graphic object layout with interactive genetic algorithms. In: Proc. of IEEE Workshop on Visual Languages, pp. 74–87 (1992)
6. Peñalver, J., Guervós, J.: Optimizing web page layout using an annealed genetic algorithm. In: Proc. Parallel Problem Solving from Nature V, pp. 1018–1027 (1998)

7. Takagi, H.: Interactive evolutionary computation: Fusion of the capabilities of ec optimization and human evaluation. Proceedings of the IEEE 89, 1275–1296 (2001)
8. Dorigo, M., Schnepf, U.: Genetics–based Machine Learning and Behaviour Based Robotics: A New Synthesis. IEEE Transactions on Systems, Man and Cybernetics 23(1), 141–154 (1993)
9. Becker, A., Seshadri, M.: Gp-evolved technical trading rules can outperform buy and hold. In: Proc. of the 3rd Int.Workshop on Computational Intelligence in Economics and Finance (2003)
10. Kay, J., Kummerfeld, R., Lauder, P.: Managing private user models and. In: Brusilovsky, P., Corbett, A.T., de Rosis, F. (eds.) UM 2003. LNCS, vol. 2702, Springer, Heidelberg (2003)
11. Milani, A., Marcugini, S.: An architecture for evolutionary adaptive web systems. In: Deng, X., Ye, Y. (eds.) WINE 2005. LNCS, vol. 3828, pp. 444–454. Springer, Heidelberg (2005)