Intelligent Information Management, **2010, 2, 597-607**
doi:10.4236/iim.2010.210068 Published Online October 2010 (http://www.SciRP.org/journal/iim)

A Min-Conflict Heuristic-Based Web Service Chain Reconfiguration Approach

Haifeng Li[1,2], Xiaoxia Yang[2]
[1]*School of Civil Engineering and Architecture,Central South University, Changsha, China*
[2]*State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, Wuhan, China*
E-mail: *lehaifeng@gmail.com*
Received April 16, 2010; *revised July* 10, 2010; *accepted September* 15, 2010

Abstract

The state-of-art Web services composition approaches are facing more and more serious bottlenecks of effectiveness and stability with the increasing diversity and real-time requirements of applications, since new web service chain must be generated from "scratch" for each application. To break these bottlenecks, this paper presents a Min-Conflict Heuristic-Based Web Service Chain Reconfiguration Approach(MCHRC) to maximal reuse relative web services chain: a min-conflict heuristic based regression search algorithms is proposed to implement the web services chain reconfiguration based on the formal definition of process constraint and integrity constraint to guarantee the correctness and integrality of the reconfiguration. This benefits the service reuse and then can relieve the time complexity of web service composition and improve web services chain executing stability by reduce service provider load. Experimental results show that this approach makes significant improvement on the effectiveness of web services composition.

Keywords: Web services composition, Min-Conflict Heuristic, Reconfiguration

1. Introduction

Under dynamic Internet environment, as a result of the corresponding change of service chain produced by frequent change of users' demand, it inevitably has become a hard problem in respect of service-oriented computer (SOC) how service chain achieve automatic reconfiguration by efficiently adjusting to dynamic change in demand. The Web services chain reconfiguration faces on the one hand search efficiency problem resulting from "composition explosion", on the other hand problems in correctness caused by violating process constraint and integrity of original service chain because of adding and modifying operation for single service during the process of service chain reconfiguration.

AI planning and semantic based approaches [1-3] solve the automatic Web services composition problem in limited scale under close world hypothesis and determinate environment, but do not take into account how to reuse former Web services chain to satisfy the new user demand; In [4], a linear programming approach is presented to resolve Web services chain reconfigure under services' QoS change, however no user demand change is considered; In [5], a case-based reasoning method is introduced to solve Web services chain reuse, and study the matchmaking between similar user demand and Web services chain on emphases, while no reconfiguration approach which reuse very single Web service in Web services chain is considered.

In all these studies, the reuse of service chain is not taken into account upon the change of demand of users, which lead to bottleneck in performance and stability in the process of composition. Only with dynamic traits of service's QoS taken into consideration, the dynamic service composition still is unable to handle the situation if users' demand changes. Aimed at such problems, we present a min-conflict heuristic based regression search algorithms to implement web services chain reconfiguration, based on process constraint and integrity constraint. Our approach can relieve time complexity of web service composition and improve web services chain executing stability by reducing service provider load. The contributes of the paper lie in the following aspects:

1) Presents the notion of Web services chain reconfiguration and implements it by regression search algo-

rithms.

2) Formalized definition of process constraint and integrity constraint to ensure correctness and integrity in the process of services chain reconfiguration.

3) Definition of mini-conflict heuristic function to improve efficiency of search algorithms.

The rest of this paper is organized as follows: Section 2 formalized definition of basic concept and reconfiguration of service chain. Section 3 describes the definition of process constraint and integrity constraint in the Web services chain Reconfiguration. Section 4 explains min-conflict heuristic based regression search algorithms in detail. Section 5 presents the results of experiments and a comparison between our approach and WSPR [6]. The related work is discussed in section 6. The paper concludes in section 7 by summarizing the contributions of this novel technique and describing a few ideas for future work.

2. Basic Conception and Definition

2.1. Real-World Scenario

Consider user Z want to spend his holiday in Y resort in X city, but he has to take part in a conference temporarily (supposed that the resort and the meeting in the same city). Those tasks involve several services as shown in **Figure 1**.

Book flight service w_1 gives flight ticket order service. It takes destination city, departing date and identity ID as preconditions and arrival date, flight NO as effects. Finding hotel service w_2 finds a hotel according to use preference. It takes destination address, and use condition as preconditions and hotel name, hotel address and hotel zip code as effects.

Booking hotel service w_3 orders hotel for use. It takes arrival date, identity id and hotel name as preconditions and room number, hotel order as effects.

Travel agent service w_4 gives guide information and presale resort ticket services for use. It takes destination address, flight no, hotel order form as preconditions and guide id, resort ticket as effects (supposed that the agent need use to provide evidences to protect his business).

Conferences register service w_5 logins conference for use. It takes flight NO, hotel order form as preconditions and registration id, conference program as effects.

The Web services chain: w_1 - w_2 - w_3 - w_4, can meet use travel requirement, as shown black and blue lines in **Figure 1**. The left of every service is preconditions and the right of it is effects. To satisfy the second conferences demand, we do not generate a new services chain from "scratch", but reuse and modify previously generated services chains. The new services chain will be w_1 - w_2 - w_3 - w_5. Thus, services w_1, w_2, w_3 is reused, and service w_4 is modified to w_5.

Therefore, we proposal a Web services chain reconfiguration approach to locally repair previously services chains to satisfy new use demands.

2.2. The Conception of Web Services Chain Recon-Figuration

The essence of SOC is the collection of information [7], which collect available service resource on Internet and

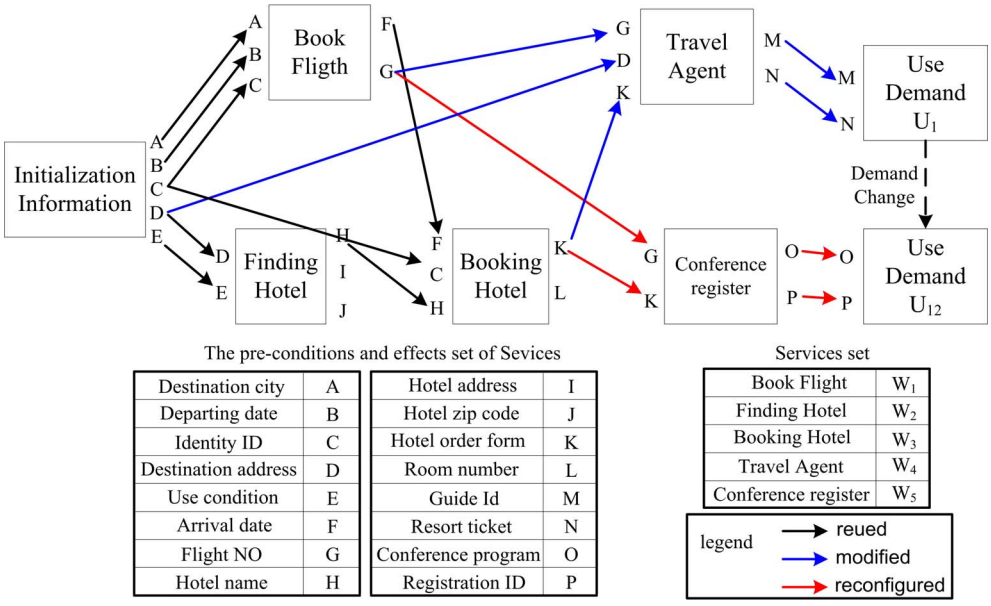


Figure 1. The conception of Web services chain reconfiguration.

information resource to achieve value-adding of service. Consequently, service composition problem is described as finding solutions to a service chain to satisfy given users' demand within finite information state space.

Definition 1: Web services chain reconfiguration is defined as 4-tuple $Recon = \langle Comp^o, G^o, Comp^d, G^d \rangle$: Given: 1) a target Web service composition problem $Comp^d$ 2) an initial composition problem $Comp^o$ and corresponding Web service chain G^o, resolve corresponding service chain G^d of $Comp^d$ based on minimal modification of G^o. Web services composition is presented as 4-tuple $Comp = \langle G, W, S, R, \rangle$ where:

G is web service chain;

W is finite nonempty set;

S is finite nonempty set of information states space which including users' input and demand information, preconditions and effects of service. States are described by ground term. There is finite change of state space produced by the execution of every service, so in order to solve so-called frame problem [8], suppose that information state set without change still stay the same.

R is presented as binary $\langle R^{in}, R^{out} \rangle$ to describe the user demand, including $R^{in} \subset S$ describe the input information states of user, and $R^{out} \subset S$ describe the information output of user demand.

As **Figure 2** shows, a new Web services chain satisfy new user demand is achieved by similar services chain local reconfiguration under the process constraint and integrity constraint. The key problem how to get G^o is a services chain matchmaking problem according to the relativity of user demand, which is beyond the scope of this paper. In this paper, suppose that the initial services chain has already existed.

Definition 2: information states space Γ is described as a 4-tuple $\Gamma = \langle S, s_0, s_g, T \rangle$, where:

$s_0 \subset S$ is initial states of information states space, which states are true. According to frame axiom, suppose that other information states are false; $s_g \subseteq S$ is the goal states (user demand) must be satisfied $s_g = R^{out}$;

T is state transition function $T: S \times W \to S$,describe states space changed service.

Definition 3: a service $w_i \in W(G)$ is presented as $\langle N, Pcon, Eff \rangle$ by 3-tuple, where:

N is the name of service, unique identifier of service w_i,

$$Pcon(w_i) = \{ pcon_j \mid pcon_j \in S, j = 1, \cdots, n \}$$

describes necessary preconditions set before service w_j execution, Eff : $Eff(w_i) = \{ eff_j \mid eff_j \in S, j = 1, \cdots, n \}$, describes change in information states space after service w_i execution.

Definition 4: Service chain is defined as a graph $G\langle W, E \rangle$,of which $W(G) = \{w_1, \cdots, w_n\}$ refers to set of services, $E(G) = \{e_1, \cdots, e_m\}$ describe constraints between services, $e_k = \{ \langle w_i, w_j \rangle \mid e_k \in E(G), w_i, w_j \in W(G) \}$ refers to combinable between two services.

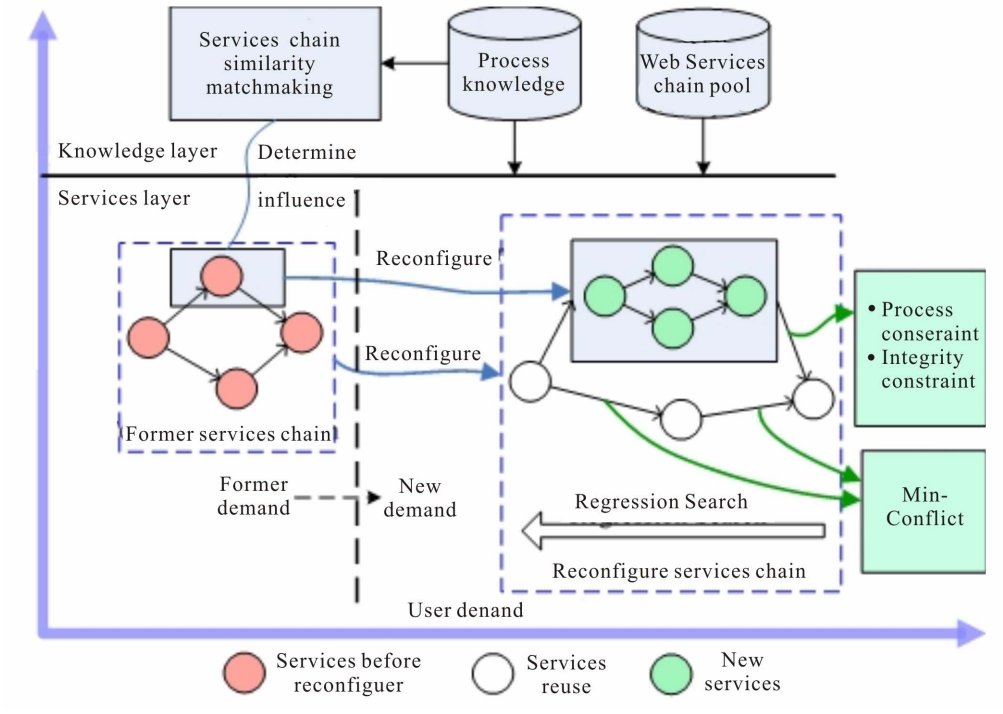


Figure 2. The conception of Web services chain reconfiguration.

Definition 5: Information state-service mapping function $f: S \to W$, $f(s) = w_i, s \in S, w_i \in W$. f is injective function, since, to any instance of information states $s \in S$, belongs to only one service.

3. Formal Definition of Process Constraint and Integrity Constraint

When user demand changes, it is needs to adaptive reconfigure some services in the relative Web services chain, which means some component services may be reused, and some may be added, deleted and modified in order to adjust to the demand change. In this dynamic local revision, the Web services chain should maintain correctness and integrity.

Correctness requires that correctness of business process can't be violated by Web services chain's reconfiguration. That is to say, some services are restricted to certain process with fixed execution conditions which presented as strict sequence relation. In the procedure of service chain reconfiguration, it presents that the preconditions of a service can't be satisfied by the effects of successor services in business process. As the same logic goes, the effects of a service can't satisfy the preconditions of predecessor service in process logic.

Integrity means that the integrity of service preconditions set must be keep in the reconfiguration. The necessary condition of the execution of service is the all the preconditions of Web service are satisfied. In the process of service chain reconfiguration, the pre-conditions of a service can't be satisfied as other services are deleted or replaced. For example, the number of the preconditions of service w_i is 3, but with some service w_j deleted, the satisfied preconditions of w_i changes into 2, thus breaking integrity constraint of preconditions of w_i. In this case, Web services chain can't work properly.

3.1. Process Constraint

Process constraint of Web services chain reconfiguration is the strict sequence relation in the business process. This constraint defines strict order logic in procedure which once being against, mistaken procedure will appear.

Definition 6: process constraint is presented as binary relation $\prec$ in service set W, $w_1 \prec w_2$, iff w_1 executes before w_2, $w_1, w_2 \in W$. Process service set $W_{prior} \subseteq W$, $\forall w_i, \exists w_j \in W_{prior}$ such that $W_{prior} = \{w_i \mid w_i \prec w_j \ or \ w_j \prec w_i\}$.

Theorem 1: W_{prior} and binary relation $\prec$ defined above make up partial ordered set $\langle W_{prior}, \prec \rangle$.

Definition 7: transitive relation in W_{prior} is defined with adjacency matrix $A = (a_{ij})$, $a_{ij} = \begin{cases} 1 \ if \ \exists w_i \prec w_j \\ 0 \ else \ or \ i = j \end{cases}$, if $\prec$ relation exist between w_i and w_j is 1, otherwise it is 0. For the purpose of convenience in calculation, suppose that self-relation of w_i is 0. Any transitive relation of two services can be calculated via transitive closure and transitive matrix corresponding.

Definition 8: for $w_i \in W_{prior}$, w_i predecessor services set $PreW_{prior}(w_i) = \{w_j \mid a_{ji} \neq 0, j \neq i\}$, w_i successor services set $RearW_{prior}(w_i) = \{w_j \mid a_{ij} \neq 0, j \neq i\}$.

Predecessor service set of a service refers to the services must be executed before a service and successor services set of service refers to the services only can be executed after service. Hence, the preconditions from service's predecessor services can't be satisfied by $Eff(w_i)$, as the same logic goes, the effects from service's successor services can't satisfy the $Pcon(w_i)$, which is shown as **Figure 3**. For instance, a service is to search the names of books on web, with its preconditions as names and ISBN, while the effects of a service of delivering books is also names and ISBN. The service to search the names of books must happen before delivering the books, thus the effects of delivering the books can't satisfy the preconditions of searching the books on web, which against the process constraint.

Definition 9: the available effects which $\forall w_i \in W$ relative to $\forall w_j \in W$ is:

$$EffS_{avail}(w_i, w_j) = \begin{cases} Eff(w_i) \setminus PconS_{void}(w_i) \ if \ w_j \in PreW_{prior}(w_i) \\ Eff(w_i) \qquad\qquad else \end{cases}$$

the available preconditions which $\forall w_i \in W$ relative to $\forall w_j \in W$ is:

$$PconS_{avail}(w_i, w_j) = \begin{cases} Pcon(w_i) \setminus EffS_{void}(w_i) \ if \ w_j \in RearW_{prior} \\ Pcon(w_i) \qquad\qquad else \end{cases}$$

3.2. Integrity Constraint

Adding or revising services in service chain reconfiguration result in that the preconditions of the others services can't be satisfied. As a result, it becomes very important to maintain the integrity of every service in the process of reconfiguration. Integrity constraint also means combinable between services, which comes from Causal Theory [9]. The preconditions of service denote the cause,

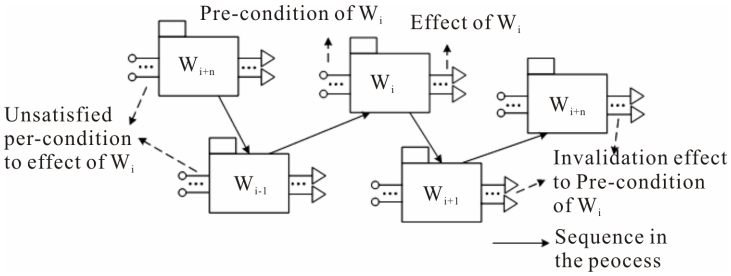


Figure 3. Process constraint.

and the effects of service denote result with the service executing, which offers basis theory for combinable between services.

Definition 10: Matchmaking states set of two services is $Match(w_i, w_j) = EffS_{avail}(w_i, w_j) \cap PconS_{avail}(w_j, w_i)$.

Definition 11: w_i, w_j are $combinable(w_i, w_j)$, *iff* $Match(w_i, w_j) \neq \varnothing$, *i.e.* the combinable of two services that the available effects of former service should satisfy the available preconditions of the later service. We deal with the available preconditions and effects on the foundation of process constraint.

Definition 12: Two states are matched denote as $Match(s_i, s_j)$, *iff* $combinable(f(s_i), f(s_j))$ and $s_i = s_j$, matching state sets of two states set defined as $MatchNum(S_1, S_2)$.

Definition 13: w_i is satisfied, presented as $satisfied(w_i)$, *iff* $\forall pcon \in Pcon(w_i)$, *s.t.* $pcon \in \bigcup_{w_j \in G} EffS_{avail}(w_j, w_i)$, *i.e.* the preconditions of w_i is satisfied by the effectiveness of w_j. If w_i is satisfied, then w_i is of constraint integrality.

Definition 14: Full match. w_i full match w_j, *iff* $EffS_{avail}(w_i, w_j) \supseteq PconS_{avail}(w_j, w_i)$.

Definition 15: Partial match. w_i partial match w_j, *iff* $\exists pcon \in Pcon(w_j)$, *s.t.* $pcon \notin EffS_{avail}(w_i, w_j)$.

If all the services in service chain can reach full match, the chain can form in a sequence which regarded as single-source path planning problem. If partial matching and full matching coexist, the problem can be cast to a proposition *STRIPS* [10], whose complexity is proved as *NP*-complete [6].

4. Min-Conflict Heuristic Based Regression Search Algorithms

Web services chain reconfiguration generate new services chain satisfy new user demand based on reuse former one. There are much redundancy searches which will lead to inefficiency in forward search since the former one has partial satisfy the user demand. What's more, for huge initial states space, forward search is difficult to converge to correct solution fleetly in finding solutions. Hence, this paper introduces a Min-Conflict Heuristic Web Service Chain Reconfigure (MCHRC) approach to achieve reconfiguration via regression search. The min-conflict heuristic function guide search process to promise search away from "combination explore" problem. Process constraint and integrity constraint are considered in the regression search to assure the correctness of the services chain reconfiguration.

4.1. Regression Search Algorithms

Regression search algorithms start from goal (user demand) and stop at user demand and services in the services chain are satisfied. The sub goals $s_{subgoal}$ are the user demand and preconditions set of services that are not satisfied in searching. The main process is divided as follows: initial, service matching, min-conflict heuristic function, maintaining search states space.

Initial: initialize the sub-goal and decide whether the sub-goal is satisfied by the former services chain. If satisfied, reuse the present service chain. Otherwise, choose the user demand cannot be satisfied by the current service chain through $s_g \setminus s(G^o)$. New service should be selected to satisfy those sub goals.

Min-Conflict Heuristic function: based on $SAT_{subgoal}$, select a service ω which minimal conflict with the current state and add it to the current service chain G^o.

Maintaining search states space: as ω is added, a new state is introduced: the preconditions and effects of ω. The unsatisfied preconditions of ω should be added as sub goals and the effects of ω is added to the effects set s_{eff} and the sub goals be satisfied be removed

Iterate the process mentioned above, until all sub goals are satisfied. Finally, a new services chain is extracted from G^o, since the former services chain has exist in G^o, the reconfiguration has been achieved. The process shows as **Figure 4** in detail.

Globe Variable: W finite service set

S finite information space set

Function $RegressionSearch(s_0, s_g, R^{in}, G^o)$ return G^d

Initial: s_0 //initial information of states space

s_g // goal states information of user demand

R^{in} // states information of user input

G^o // initial service chain

$s(G^o) \leftarrow \bigcup_{w_i \in G^o} Eff(w_i) s_0 \cup R^{in}$

$s_{eff} \leftarrow s_g \setminus s(G^o)$

$s_{subgoal} \leftarrow s_{eff}$

$while(\neg s_{subgoal})$

$SAT_{subgoal} \leftarrow find\_service(W \setminus G^o, s_{subgoal})$

$\omega \leftarrow min\_conflict(SAT_{subgoal}, s_{subgoal}, s_{eff})$

$G^o \leftarrow G^o \cup \{\omega\}$

$maintenance\_state(\omega, s_{eff}, s_{subgoal})$

$G^d \leftarrow create\_graph(G^o)$ //

End Function

Figure 4. Regression search algorithms

4.1.1 The Best Result

The best result of Web services chain reconfiguration is that service chain requires no revising, satisfying the new demand directly. This means that former services chain is able to satisfy new user demand without any change. The time complexity of algorithms is $O(G^o \times n_w)$ can be finished in linear time.

4.1.2 The Worst Result

The worst result of algorithms is that no services can be reused which degenerate to Web services composition problem. The algorithms in this paper temporarily don't deal with *k-SAT* problem including in service matchmaking between preconditions and effects. When $k \geq 3$, it is *NP*-Complete[11]. So the services chain reconfiguration with *k-SAT* problem is at least *NP*-Complete.

4.2 Mini-Conflict Heuristic Function

Min-conflict heuristic strategy is design for reduce the blindness of searching. Mini-conflict heuristic theory is first introduced by [12], applied to solve of constraint satisfying and schedule problem by local revising. If a new service is added to the current services chain, it will satisfy the sub goals to reduce the number of unsatisfied sub goals. But at the same time, the preconditions of this service also increases the number of the sub goals that should be satisfied. So the key of min-conflict is to reuse the effects in the current to eliminate the inconsistent between sub goals and effects. Hence, min-conflict could be defined in two facets as following: shown as **Figure 5** and Min-conflict heuristic function shown as **Figure 6**.

1) The maximal satisfied effects refers to new services can maximally satisfy sub goals, *i.e.* using the least services to satisfy the most sub-goals.

2) The minimal violating preconditions means that the preconditions of new service are satisfied maximally *i.e.* minimize the new sub goals added by preconditions of new service.

Min-conflict heuristic is a depth first local optimality strategy, so there is possibility to guide the search to wrong branch and lead the failing of the solve process. To avoid it, we use depth limited based backtracking [13], and backtracking when: 1. the depth is larger than threshold; 2. the preconditions cannot be satisfied by any states in state space. Depth limited based backtracking is a classic method, we don't describe here in detail for limited length.

4.3. Maintaining Search States Space

As the service in the current services chain can't satisfy the user demand, it is required that new service should be added to satisfy demand again. The principle of reconfiguration is what kind of services can be reused to satisfy the new user demand and what kind of services should be added for unsatisfied sub goals.

In the regressed search these two kinds of states information should be maintained. The first one is the available service effects to satisfy the user demand and preconditions of service which including reusable services. The later one is to maintain the sub goal to be satisfied, which include delete the sub goals have been satisfied and add the preconditions of services have not been satisfied yet. Regressed search will stop until the sub goal set is empty. The process is shown as **Figure 7**.

5. Experiment and Analyze

In our experiment WSPR [6] is chosen as object to compare, since regression search is used by both, what's

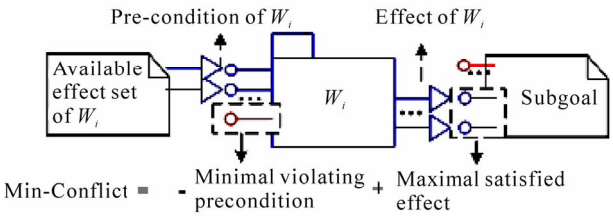


Figure 5. The definition of min-conflict.

Function $min\_conflict(SAT_{pcon}, s_{pcon}, s_{eff})$ reture ω
  $min\_conflict \leftarrow 0$
  for each w_i *in* SAT_{pcon}
    $max\_sat \leftarrow |MatchNum(Eff(w_i), s_{subgoal})|$
    $min\_pre \leftarrow |MatchNum(s_{eff}, Pcon(w_i))|$
    if $(max\_sat - min\_pre > min\_conflict)$ *then*
      $min\_conflict \leftarrow max\_sat - min\_pre$
      $\omega \leftarrow w_i$
End Function

Figure 6. Min-conflict heuristic function.

Function $maintenance\_state(\omega, s_{eff}, s_{subgoal})$
  $s_{eff} \leftarrow s_{eff} \cup Eff(\omega)$
  $s_{subgoal} \leftarrow s_{subgoal} \setminus Mach(s_{subgoal}, Eff(\omega))$
  $s_{subgoal} \leftarrow s_{subgoal} \cup \overline{Mach(Pcon(\omega), s_{eff})}$
End Function

Figure 7. The process of maintaining search states space.

more WSPR is first runner-up Award of Web services composition challenge at IEEE CEC/EEE(ICEC) which ensure that the experiment is representational.

5.1. Dataset and Experiment Scheme

Data set choose. We use the data set from syntax part of ICEC Web services Composition Challenge, since the semantic part is described by WSDL while not OWL-S. And we only show the Composition1 data set of syntax part because of length limited here. There are nice services set with difference by the number of services and preconditions and effects corresponding. For example Composition1-20-4 means that there are 2156 services and 4 preconditions and effects respectively. For more information refer to literature [6].

Experiment scheme. Our method is design for the Web services chain reconfiguration for new user demand. Since ICEC data set do not provide services chain, we use method as following to simulate a former services chain: we first generate a Web Services chain through WSPR, from which 50% web services is chosen as reused services. These services are the basic of services chain reconfiguration. We also construct a partial order services set via random sample, and make a transitive matrix as process constraint by Warshall Algorithms [14] at initialization.

5.2. Experiment Result and Analyses

The experiment performs on 9 sub services set and 11 queries corresponding each sub services set in Composition1. The Service chain generation time for each query is shown as **Table 2** Composition1-20,50,100 means that there are 2156, 2656, 4156 services in the services set respectively and 4,16,32 means that there are 4,16,32 preconditions and effects in each services set respectively. The services reuse and iterate times in reconfiguration for each query are shown as **Table 2**. The compare of WSPR and MCHRC are shown as **Figure 8**, **Figure 9**, **Figure 10**. The result of data set Compositon1-20 (Compositon1-50 and Compositon1-100 is the same) is shown as **Figure 8**, in which figure a),figure b), figure c) show the difference between the number of preconditions and effects. As the number increase, MCHRC and WSPR all spend more time on the matchmaking of the services. The efficiency difference of MCHRC and WSPR is become smaller from figure a) to figure c), because the process constraint and integrality constraint are considered in MCHRC, while WSPR not. So WSPR is unreliable to the services set which strict sequence order is defined. Compare with figure a) in the figures, it will be found that MCHRC is insensitive to the number of services, because we create index on services set to accelerate the search process to find a single service which inspired by [15]. It will also be found that the 11^{th} query cost very little time, because exist a services chain to satisfy the new user demand. This is the best situation that reusing all the services from former services chain.

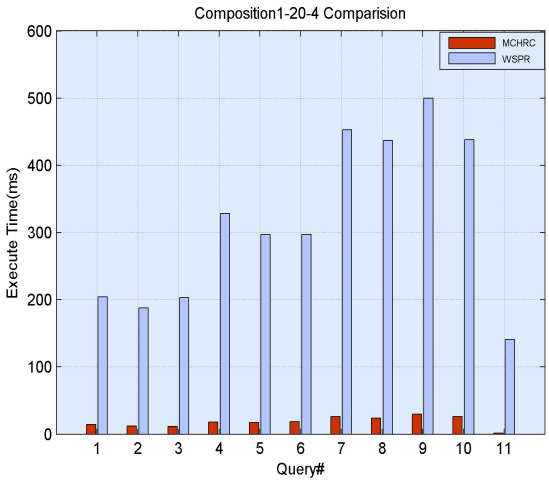
The result shows that the sharp increase in efficiency contrast MCHRC with WSPR, there are two main reasons:

The services reuse. As **Table 1** shows, MCHRC keep some reusable services in reconfiguration which decrease the search times effectively, for instance, the 11^{th} query.
Min-conflict heuristic strategy. In the test data, as shown in **Table 1**, we quickly find solution almost without any backtracking because the Min-conflict heuristic guides the search effectively. The reason is that the distribution of solution of test data is very fit for the min-conflict heuristic strategy, just as the resolve million queens problem just in one minute [16].

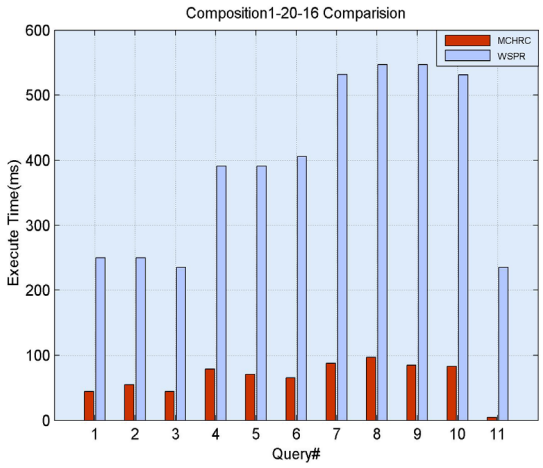
6. Related Works

The related works focus on state-of-art Web services composition, since little literature discusses the Web services configuration. There are two kinds of method in current research: the static and dynamic Web service composition.

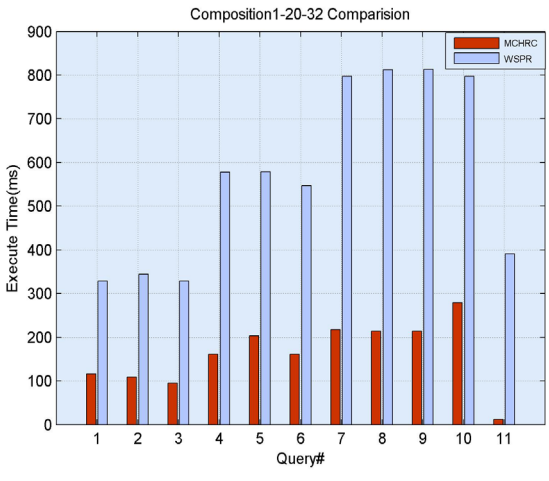
Static Web service composition. Static service composition refers to pre-defined environment model which suppose all the information in this model are unchangeable, the composition remains static and accurate during it life periods. The actually industrial standard, BPEL4WS,



(a)

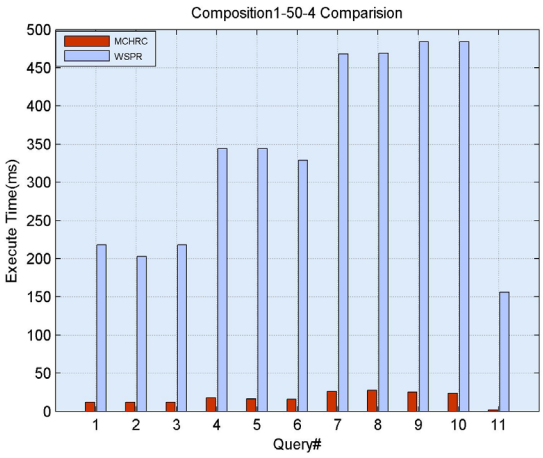


(b)

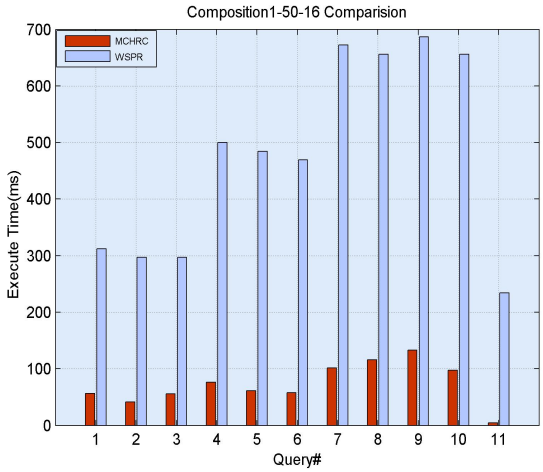


(c)

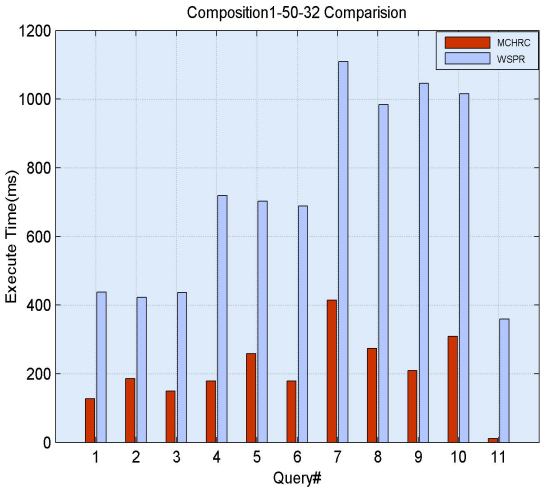
Figure 8. Efficiency compare in Composition1-20 set. (a) Efficiency compare in 20-4 set; (b) Efficiency compare in 20-16 set; (c) Efficiency compare in 20-32 set.



(a)

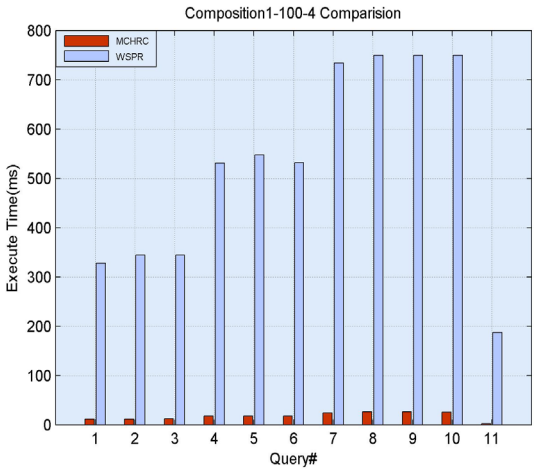


(b)

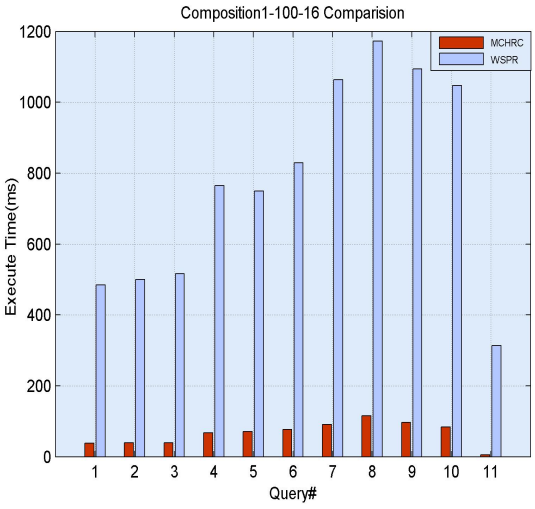


(c)

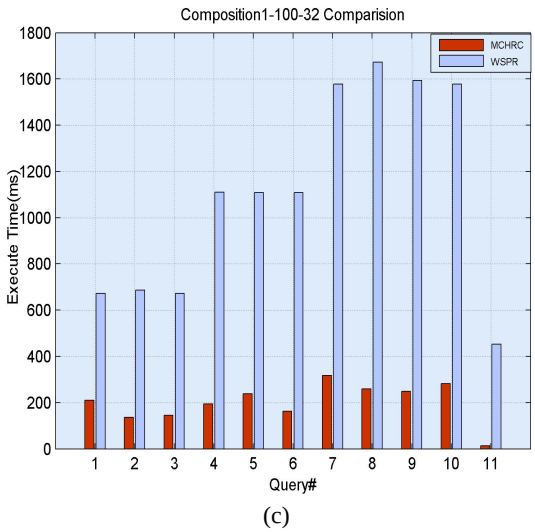
Figure 9. Efficiency compare in Composition1-50 set. (a) Efficiency compare in 50-4 set; (b) Efficiency compare in 50-16 set; (c) Efficiency compare in 50-32 set.



(a)



(b)



(c)

Figure 10. Efficiency compare in Composition1-100 set. (a) Efficiency compare in 100-4 set; (b) Efficiency compare in 100-16 set; (c) Efficiency compare in 100-32 set.

pre-definite whole service chain process as templates, results in unfavorable flexibility of modification [17]. In [18], a forward search approach is introduced to achieve composition of Web services, but it will lead to low efficiency of search. The main principle of automatic service composition is based on the approach of AI planning which cast service composition to searching of states and services space. A typical example is HTN-based method which is applied to achieve ser- vice composition through building up mapping from OWL-S description service to PDDL language, adopting SHOP2 to resolve composition problem [19]. Semantic based method mainly focus on the conceptions relationship of services using description logic reason the subsume relation and consistency them [3]. But all these methods lack of reuse mechanism on Web services chain reconfiguration, hence poor in Web Services reuse.

Dynamic Web service composition. Dynamic service composition is to ensure the completion of the task and maintain the optimization of pre-defined object function by inspection on environment and replace the failed service to keep the optimization. In [4], the services chain is divided into three stages: completed, being executed and to be executed. Each time the QoS of single service changes, the method of integer linear programming is adopted to search of replaceable best solution among the service set out of completed services. In [20], a negotiate mechanism is introduced to handle the dynamic of Web services QoS change to realize re-optimal. Dynamic service composition, faces to concrete service level which is more about services' QoS. However, it is not taken into account that the users' demand changes how to reuse former services. The reuse of service can be achieved via the approach of case-based reasoning. In [5], similarity based matchmaking between two service chains is researched to find a service chain which is most relative to a new user demand. But, there is no sufficient study on how to reconfigure the services chain.

Table 1. Services reuse and search times

Query id	Service chain length	Reuse services number	Search times	Reuse ratio
1	2	1	1	50%
2	2	1	1	50%
3	2	1	1	50%
4	3	1	2	33.3%
5	3	1	2	33.3%
6	3	1	2	33.3%
7	4	2	2	50%
8	4	2	2	50%
9	4	2	2	50%
10	4	2	2	50%
11	1	1	0	100%

Table 2. The result of experiment in Composition1 data set

Service chain generate time(ms) / Query id	Composition1-20			Composition1-50			Composition1-100		
	4	16	32	4	16	32	4	16	32
1	14.46	43.992	116.283	12.151	56.691	127.154	11.46	38.542	210.39
2	11.758	54.837	108.553	11.805	41.29	185.884	11.196	38.959	136.678
3	11.47	44.359	94.887	12.19	55.845	148.86	11.865	39.449	145.803
4	17.786	78.792	160.717	17.78	76.49	178.704	17.68	67.925	193.422
5	17.14	70.835	202.766	16.503	60.892	258.435	17.911	70.464	238.756
6	18.467	65.482	160.749	16.082	57.51	178.169	17.357	77.177	163.035
7	26.048	87.338	217.387	25.925	101.443	414.819	23.524	90.38	317.944
8	23.767	97.051	213.655	27.469	115.818	273.843	25.858	115.757	258.535
9	29.674	84.938	214.146	25.276	132.754	208.866	26.483	96.174	247.865
10	26.161	83.062	279.19	23.468	97.409	308.64	25.2	83.301	281.258
11	1.337	4.376	11.93	1.557	4.768	10.704	1.849	4.901	14.005

7. Conclusions

In this paper, the service chain reconfiguration is to be realized by means of min-conflict based regression search algorithms to reach the target of maximal reuse of existing web services chain to satisfy the user demand of diversity and real-time. In regressed search, process constraint and integrity constraint are taken into account to guarantee the integrity and correctness of service chain reconfiguration. The experimental results show that the method used in this paper can satisfy the user demand when changes arise with correctness and high-efficiency. A preliminary study has been made in this paper in the service reconfiguration, and further study include demand analyses based relativity evaluate of web services chain, user preference based web services chain reconfigure, and take account of semantic matchmaking and control flow in the web services chain reconfigure.

8. Acknowledgement

The work was supported by NSFC(41001220) in China.

9. References

[1] X. Tang, C. Jiang and Z. Ding, "Automatic Web Service Composition Based on Logical Inference of Horn Clauses in Petri Net Models," *Proceedings of the* 5*th IEEE International Conference on Web Services* (*ICWS* 2007), IEEE Computer Society, 2007, pp. 1162-1163.

[2] D. Berardi, D. Calvanese and G. De Giacoma, "Automatic Service Composition Based on Behavioral Descriptions," *International Journal of Cooperative Information Systems*, 2005, Vol. 14, No. 4, pp. 333-376.

[3] M. Klusch, A. Gerber and M. Schmidt, "Semantic Web Service Composition Planning with OWLS-Xplan," *Proceedings of* 1*st International Symposium on Agents and the Semantic Web*, 2005, pp. 55-62.

[4] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam and H. Chang, "QoS-Aware Middleware for Web Services Composition," *IEEE Transactions on Software Engineering*, Vol. 30, No. 5, 2004, pp. 311-327.

[5] T. Osman, D. Thakker and D. Al-Dabass, "Semantic-Driven Matchmaking of Web Services Using Case-Based Reasoning," *ICWS* 2006: *IEEE International Conference on Web Services, Proceedings*, USA, 2006, pp. 29-36.

[6] S.-C. Oh, D. Lee and S. R. T. Kumara, "Web Service Planner (WSPR): An Effective and Scalable Web Service Composition Algorithm," *International Journal of Web Services Research*, Vol. 4, No. 1, 2007, pp. 1-23.

[7] C. T. Kwok and D. S. Weld, "Planning to Gather Information," *Proceedings of the* 13*th National Conference on Artificial Intelligence* (*AAAI*), Salt Lake City, 1996, pp. 32-39.

[8] R. B. Scherl and H. J. Levesque, "Knowledge, Action, and the Frame Problem," *Artificial Intelligence*, Vol. 144, No. 1-2, 2003, pp. 1-39.

[9] M. Prokopenko, "A Preferential Semantics for Causal Reasoning about Action," *Annals of Mathematics and Artificial Intelligence*, Vol. 46, No. 4, 2006, pp. 375-413.

[10] R. E. Fikes, and N. J. Nilsson, "STRIPS: A New Approachto Theorem Proving in Problem Solving," *Artificial Intelligence*, Vol. 2, 1971, pp. 189-208.

[11] C. Castellini, E. Giunchiglia and A. Tacchella, "SAT-Based Planning in Complex Domains: Concurrency, Constraints and Nondeterminism," *Artificial Intelligence*, Vol. 147, No. 1-2, 2003, pp. 85-117.

[12] S. Minton, M. D. Johnston, A. B. Philips and P. Laird, "Minimizing Conflicts: A Heuristic Repair Method for Constraint Satisfaction and Scheduling Problems," *Artificial Intelligence*, Vol. 58, 1990, pp. 161-205.

[13] S. Russell and P. Norvig, "Artificial Intelligence: A Modern Approach," Prentice-Hall, New Jersey, USA, 2004.

[14] K. H. Rosen, "Discrete Mathematics and Its Application," China Machine Press, China, 1999.

[15] B. Xu, T. Li, Z. Gu and G. Wu, "SWSDS: Quick Web Service Discovery and Composition in SEWSIP," *Proceedings of The* 8*th IEEE International Conference on E-Commerce Technology and the* 3*rd IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services* (*CEC/EEE*'06) *IEEE Computer Society*, 2006, pp. 71-73.

[16] R. Sosic and J. Gu, "Efficient Local Search with Conflict Minimization: A Case Study of the n-Queens Problem," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 6, No. 5, 1994, pp. 661-668.

[17] X. Fu, T. Bultan and J. Su, "Analysis of Interacting BPEL Web Services," *Proceedings of the* 13*th International World Wide Web Conference* (*WWW*'04), ACM Press, New York, 2004.

[18] S. Thakkar, C. A. Knoblock, J. L. Ambite and C. Shahabi, "Dynamically composing Web services from online sources," *Proceeding of the AAAI*-2002 *Workshop on Intelligent Service Integration*, Edmonton, Alberta, Canada, 2003, p. 1-7.

[19] E. Sirin, B. Parsia, D. Wu, J. Hendler and D. Nau, "HTN Planning for Web Service Composition Using SHOP2," *Proceedings of* 2*nd International Semantic Web Conference* (*ISWC*2003), Florida, 2003, pp. 20-23.

[20] D. Ardagna and B. Pernici, "Adaptive Service Composition in Flexible Processes," *IEEE Transactions on Software Engineering*, Vol. 33, No. 6, 2007, pp. 369-384.

