doi:10.11918/j.issn.0367-6234.2016.07.025

融合语义知识库的流程匹配算法

常关羽, 杨海成, 孙 鹏

(西北工业大学 机电学院, 西安 710072)

摘 要: 为在流程相似度计算中加入流程间深层语义关联的度量,同时在流程节点较多的情况下,实现流程匹配算法在寻优时间复杂度和相似度匹配输出值两方面的综合优化,提出一种面向流程的遗传匹配算法,将遗传算法引入并应用在流程语义和结构的相似度计算寻优过程中. 确定遗传算法的参数编码方式,并利用贪婪算法进行初始种群的设置,定义各个遗传算子,提出有效的简化策略,解决了流程节点较多时流程匹配过程寻优问题. 实验研究表明,在流程节点数较多时,本文算法在寻优时间花费和相似度值两方面的折中优化性能明显优于其他两种算法. 将遗传算法应用到流程的相似度计算及其寻优过程,可以有效地控制时间复杂度并保证较好的匹配输出结果.

关键词: 业务流程; 文本相似度; 语义知识库; 匹配相似度; 遗传算法

中图分类号: TP315 **文献标志码**: A **文章编号**: 0367-6234(2016)07-0150-06

Process matching method based on semantic repository

CHANG Guanyu, YANG Haicheng, SUN Peng

(School of Mechanical Engineering, Northwestern Polytechnical University, Xi'an 710072, China)

Abstract: To calculate the process similarity with consideration of deep semantics correlation between business processes, and to optimize the time complexity and matching result when the node number of business process becomes larger and larger, a process matching method based on GA (Genetic Algorithm) is put forward. This method is applied in similarity calculation for both process semantic and process structure, in which encoding is determined, and greedy algorithm is utilized to initialize the population of GA. By defining genetic operations and adopting some strategies for simplifying, the optimization of business process matching with large node number is fulfilled. As is expected, the experiments prove that the overall performance of algorithm proposed in this paper is better than the others that exist, especially when the count of process nodes grows to a large number. So it is concluded that the application of GA in business process similarity calculation and corresponding process optimization can effectively control the time complexity, meanwhile ensure the quality of the matching result, which shows a good practicability.

Keywords: business process; text similarity; semantic repository; match similarity; genetic algorithm

海量流程资产的有效利用取决于信息系统的流程管理技术水平,而有效的流程相似度查询技术则成为提升业务流程管理水平的关键技术之一[1]. 在相似性计算方面,标注于流程节点上的文本标签的相似度是进行流程相似性分析的基础[2-3]. 传统的基于文本的匹配检索可以用来索引和搜寻业务流程模型知识库. 但这类匹配和检索是以关键词和字符匹配[4-5]或其特征值[6]的近似程度为基础,若模型标记的文本标签包含特定的关键词或字符,那么这种方式是确有成效的[7]. 但是,这种相似的应用没有考虑语义异构性的存在. 这使得采用以关键词和字符匹配为基础的传统流程相似匹配难以实现更加准确的语义查询[8]. 基于语义知识库的语义相似度是一种计算文本与文本间在其概念关联上相似程度的度量方法. 因此,本文引入基于语义知识库的概念相似度计算方法,将其应用在流程匹配中的文本标签的相似度计算中,以提高流程匹配的准确度.

在流程匹配过程的寻优算法上,现存的几种流程匹配算法都具备各自的特点,但在流程节点较多时,都表现出各自的不足. 例如贪婪算法[9],可以在很短时间内得到一个匹配结果以及匹配的相似度

收稿日期: 2015-03-26
基金项目: 国家自然科学基金(51375395)
作者简介: 常关羽(1985—),男,博士研究生;
杨海成(1959—),男,教授,博士生导师
通信作者: 常关羽,dengxiao@mail.nwpu.edu.cn

值，但很可能得到的不是最优匹配. 而作为典型的启发式算法的 A * 算法[10]，虽然可以确保找到最优解，但其耗时代价可能会随着节点数增加而急剧增长. 因此，本文提出一种基于遗传算法[11-12]的折中优化的算法，旨在节点数较多时，既能在相似度上达到较为满意的值，又可以在匹配寻优过程的时间耗费上处于可以接受的范围.

1　节点语义相似度计算模型

1.1　语义相似度及其关键定义

定义 1　义原相似度[13]. 设 s_1 与 s_2 为两个义原，义原间距离记为 $D(s_1,s_2)$，相似度记为 $S_{\text{sem}}(s_1,s_2)$，则

$$S_{\text{sem}}(s_1,s_2)=\frac{\alpha\times(d_1+d_2)}{(D(s_1,s_2)+\alpha)\times\max((d_1-d_2),1)}.$$

其中：d_1 和 d_2 分别是义原 s_1 和义原 s_2 所处的层次，$\alpha>0$，且 α 是相似度为 0.5 时 s_1 和 s_2 之间的距离.

定义 2　义原最佳匹配[14]. 设 $S_1=\{s_{11},s_{12},\cdots,s_{1m}\}$ 和 $S_2=\{s_{21},s_{22},\cdots,s_{2n}\}$ 分别为包含 m 和 n 个义原的集合，且有 $m\leqslant n$. 定义从 S_1 到 S_2 的一个单射集合为 M. 则义原的最佳匹配是指一个单射集合 M^{opt}，对于所有其他的单射集合 M 都有

$$\sum_{(s_{1i},s_{2j})\in M^{\text{opt}}}S_{\text{sem}}(s_{1i},s_{2j})\geqslant\sum_{(s_{1i},s_{2j})\in M}S_{\text{sem}}(s_{1i},s_{2j}).$$

其中 S_{sem} 为两个义原间的相似度.

定义 3　义项相似度. 设 C_1、C_2 为两个义项，C_1 由 m 个义原描述 $s_{11},s_{12},\cdots,s_{1m}$，$C_2$ 由 n 个义原描述 $s_{21},s_{22},\cdots,s_{2n}$，且 $m\leqslant n$. M^{opt} 为描述两个义项的义原集合的最佳匹配，则 C_1、C_2 的相似度为[15]

$$S_{\text{Concept}}(C_1,C_2)=\sum_{\substack{i=1\\(s_i,s_j)\in M^{\text{opt}}}}^{m}w_i\times S_{\text{sem}}(s_i,s_j).$$

其中 w_i 是为不同的义原映射赋予的不同权值.

定义 4　概念相似度. 设概念 S_1 的名称 L_1 有 m 个义项 $C_{11},C_{12},\cdots,C_{1m}$，概念 S_2 的名称 L_2 有 n 个义项 $C_{21},C_{22},\cdots,C_{2n}$，则概念 S_1 和概念 S_2 的相似度为[16-17]

$$S(S_1,S_2)=\max_{\substack{i=1\cdots m\\j=1\cdots n}}(S_{\text{concept}}(C_{1i},C_{2j})).$$

其中 $S_{\text{Concept}}(C_{1i},C_{2j})$ 表示义项 C_{1i}、C_{2j} 之间的相似度.

1.2　流程节点的语义相似度计算

定义 5　同义词相似度[18]. 设 $l_1,l_2\in\Omega$ 为文本标签，W 为所有词语或单词的集合，$w:\Omega\rightarrow P(W)$ 为分离文本标签为单个词汇集合的函数. $s:W\rightarrow P(W)$ 可获取给定单词的同义词. 令 $w_1=w(l_1)$，$w_2=w(l_2)$，设 w_i 和 w_s 分别为相同词汇和同义词的权值. $S(l_1,l_2)$ 为 l_1 和 l_2 的语义相似度：

$$S(l_1,l_2)=[2\cdot w_i\cdot|w_1\cap w_2|+w_s\cdot(|s(w_1,w_2)|+|s(w_2,w_1)|)]/|w_1+w_2|.$$

其中 $s(w_1,w_2)=\bigcup_{w\subseteq w_1-w_2}s(w)\cap(w_2-w_1)$ 为 w_2 中 w_1 的同义词集合.

定义 6　文本标签的语义相似度. 设 $l_1,l_2\in\Omega$ 为文本标签，W 为所有词语或单词的集合，$w:\Omega\rightarrow P(W)$ 为分离文本标签成单词集合的函数. s_{word} 是基于语义知识库的一个词汇相似度函数. 令 $w_1=w(l_1)$，$w_2=w(l_2)$，M 为词汇集合 w_1 与 w_2 的词汇单射集合. M^{opt} 为使得词汇映射相似度值之和最大的映射. $S(l_1,l_2)$ 为文本标签 l_1 和 l_2 的语义相似度，即

$$S(l_1,l_2)=\frac{2}{|w_1|+|w_2|}\sum_{(w_{1i},w_{2j})\in M^{\text{opt}}}s_{\text{word}}(w_{1i},w_{2j}).$$

其中：$|w_1|$、$|w_2|$ 分别表示 w_1 和 w_2 中的词汇数，w_{1i}、w_{2j} 分别表示 w_1 和 w_2 中的一个词语.

定义 7　节点特性相似度. 设 $B_1(N_1,E_1,\tau_1,\lambda_1,\alpha_1)$ 和 $B_2(N_2,E_2,\tau_2,\lambda_2,\alpha_2)$ 是两个流程图，令 $n_1\in N_1$ 和 $n_2\in N_2$ 分别为 B_1 和 B_2 的一个节点. S 为相似度函数. 那么可以定义节点 n_1 和 n_2 特性的相似度为

$$S_{\text{att}}(n_1,n_2)=\underset{\substack{(t_1,l_1)\in\alpha_1(n_1),\\(t_2,l_2)\in\alpha_2(n_2),\\t_1=t_2}}{F_{\text{AVG}}}S(l_1,l_2).$$

特性相似度一般与其他相似度结合起来使用.

定义 8　节点类型相似度. 设有流程图 $B_1(N_1,E_1,\tau_1,\lambda_1,\alpha_1)$ 和 $B_2(N_2,E_2,\tau_2,\lambda_2,\alpha_2)$，令 $n_1\in N_1$ 和 $n_2\in N_2$ 分别为 B_1 和 B_2 的节点. 令 $t:T\times T\rightarrow[0,1]$ 为节点类型的相似度函数，则

$$S_{\text{typ}}(n_1,n_2)=t(\tau_1(n_1),\tau_2(n_2))$$

为节点类型相似度函数. t 是预先定义的. 可选择以下参考定义形式：

$$t(t_1,t_2)=\begin{cases}1, & \text{if } t_1=t_2;\\0, & \text{else};\end{cases}$$

$$t(t_1,t_2)=S_{\text{sem}}(t_1,t_2).$$

综上，本文设计了如图 1 的节点相似度计算模型.

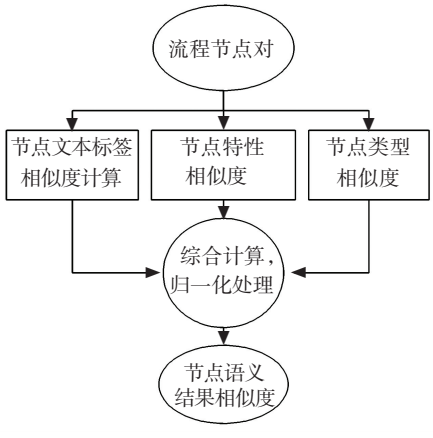


图 1　节点相似度计算模型

2 流程相似度计算模型

第 2 种要研究的相似度是业务流程的结构相似度. 结构相似度基于图编辑距离来定义[19].

定义 9 图的编辑距离. 设 $B_1(N_1,E_1,\tau_1,\lambda_1,\alpha_1)$ 和 $B_2(N_2,E_2,\tau_2,\lambda_2,\alpha_2)$ 是两个流程图, S 为相似度函数, $M:(N_1 \nrightarrow N_2)$ 为一个局部单射. $n \in (N_1 \rightarrow N_2)$ 为一个节点. 当且仅当 $n \in dom(M)$ 或 $n \in cod(M)$ 时, n 是被"替换"的. 若 n 不是被"替换"时,则 n 为"插入"或"删除". s_n 为所有"插入"或"删除"的节点集. $(n,m) \in E_1$ 为边,当且仅当不存在 $(n,n') \in M$ 或 $(m,m') \in M$ 以及 $E_{\text{edge}}(n',m') \in E_2$. s_e 是所有"插入"或"删除"的边集. 图编辑距离表示如下:

$$|s_n| + |s_e| + 2 \cdot \sum_{(n,m) \in M} (1 - S(n,m)).$$

图的编辑距离是由两个流程导出的最小化距离. 其计算方法是:1 减去"插入"或"删除"节点"插入"或"删除"边集以及"替换"节点平均距离的平均值的分数部分的差.

定义 10 图编辑距离相似度. 设 $B_1 = (N_1,E_1,\tau_1,\lambda_1,\alpha_1)$ 和 $B_2 = (N_2,E_2,\tau_2,\lambda_2,\alpha_2)$ 为两个流程图, S 为相似度函数,令 $M:(N_1 \nrightarrow N_2)$ 为导出两个流程图编辑距离的局部单射, s_n 和 s_e 同定义 9 中所述,图的编辑距离相似度为

$$\begin{cases} s_{\text{ged}}(B_1,B_2) = 1 - avg(s_{nv},s_{ev},s_{bv}), \\ s_{nv} = \dfrac{|s_n|}{|N_1| + |N_2|}, \\ s_{ev} = \dfrac{|s_e|}{|E_1| + |E_2|}, \\ s_{bv} = \dfrac{2 \cdot \sum\limits_{(n,m) \in M} (1 - S(n,m))}{|N_1| + |N_2| - |s_n|}. \end{cases}$$

定义 11 等价映射和最优等价映射. 设 $B_1(N_1,E_1,\tau_1,\lambda_1,\alpha_1)$ 和 $B_2(N_2,E_2,\tau_2,\lambda_2,\alpha_2)$ 为两个流程图,设节点对相似度函数为 $S: N_1 \times N_2 \rightarrow [0,1]$. 一个局部单射 $M_S:(N_1 \nrightarrow N_2)$ 为等价映射的条件是,当且仅当对于所有的 $n_1 \in N_1$ 和 $n_2 \in N_2$: $(n_1,n_2) \in M$ 可使 $S(n_1,n_2) > 0$.

最优等价映射 $M_S^{\text{opt}}: N_1 \nrightarrow N_2$ 是指对于所有其他的等价映射 M_S 都有

$$\sum_{(n_1,n_2) \in M_S^{\text{opt}}} S(n_1,n_2) \geqslant \sum_{(n_1,n_2) \in M_S} S(n_1,n_2).$$

定义 12 流程匹配相似度. 设 $B_1 = (N_1,E_1,\tau_1,\lambda_1,\alpha_1)$ 和 $B_2 = (N_2,E_2,\tau_2,\lambda_2,\alpha_2)$ 为两个流程图. 设 S 为计算两个节点间相似度的函数,且 t_s 为应忽略的节点类型集合. 设 $M_S^{\text{opt}}: N_1 \nrightarrow N_2$ 为通过相似度函数 S 导出的最优等价映射,其中忽略了 t_s 中的类型. 那么 B_1 和 B_2 间的节点匹配相似度为

$$S_{\text{nm}} = (2 \cdot \sum_{(n,m) \in M_S^{\text{opt}}} S(n,m)) / (|\{n \mid n \in N_1, \tau_1(n) \notin t_s\}| + |\{n \mid n \in N_2, \tau_2(n) \notin t_s\}|).$$

基于以上定义,流程相似度计算模型如图 2.

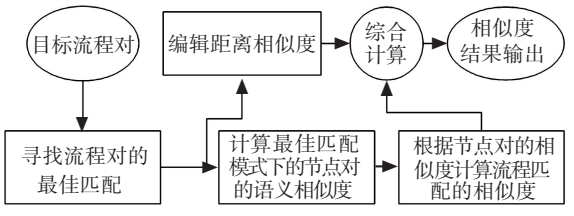


图 2 流程相似度计算模型

3 流程匹配过程寻优算法设计

本文设计了一种基于遗传算法的匹配过程寻优算法,使得计算效率和匹配效果都能满足应用需求. 算法关键设计细节如下.

3.1 流程编码

设一个有 n 个节点的流程 P_n, 采用编码 $1,2,\cdots,n-1,n$ 对每个节点进行编码. 设另一个有 m 个节点的流程 P_m, 则其编码为 $1,2,\cdots,m-1,m$. 任意两个流程的节点数总有 $n \geqslant m$. 因为,总可用 n 表示节点数较多的流程节点数. 当两个节点数分别为 n 和 m 的流程 P_n 和流程 P_m 进行匹配时,编码的位数便为 n, 编码每一个基因位上的取值范围是 $1 \sim n$, 且每一个基因位上的取值不重复. 编码中每个基因位代表 P_n 的一个节点,而该基因位的取值代表 P_m 中的一个节点. 注意到 $n \geqslant m$, 因此在基因位的取值中,超过 m 的值 $m+1,\cdots,n-1,n$ 是没有意义的. 因此,当某基因位的取值为 $m+1,\cdots,n-1,n$ 中的一个时,表示该基因位代表的流程 P_n 的节点不能与 P_m 中节点匹配.

3.2 种群初始化

首先,可以确定一个表示节点间的相似度的矩阵 $\boldsymbol{S}$. 以 P_9 和 P_7 为例,则有相似度矩阵 $\boldsymbol{S}_{9\times7}$ 为

$$\boldsymbol{S}_{9\times7} = \begin{array}{c} \\ 1 \\ 2 \\ \vdots \\ 9 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & \cdots & 7 \end{array} \\ \begin{bmatrix} s_{11} & s_{12} & \cdots & s_{17} \\ s_{21} & s_{22} & \cdots & s_{27} \\ \vdots & \vdots & s_{ij} & \vdots \\ s_{91} & s_{92} & \cdots & s_{97} \end{bmatrix} \end{array}.$$

式中, s_{ij} 为 P_9 的第 i 个节点和 P_7 中的第 j 个节点的相似度值.

可以采用贪婪算法生成初始种群. 在上述相似度矩阵 $\boldsymbol{S}$(即 $\boldsymbol{S}_{9\times7}$) 中,找到值最大的一个 s_{ij} ,然后,

将该值的列号填写到基因位的第一位即得到 $(j, \times\times\times\times\times\times\times\times\times)$，× 表示待定. 然后划掉第 i 行和第 j 列的所有相似度值，得到一个新的相似度矩阵 $\boldsymbol{S}'$，然后再寻找 $\boldsymbol{S}'$ 中值最大的元素重复以上过程，直到最后的 $\boldsymbol{S}'$ 矩阵的列数为 0，便得到了一个个体. 如果列数不等于行数，得到的个体可能为 (7,3,1,6,2,5,4, ××)，其中有两个元素不确定. 这时将行中未出现的编号填充上去得到 (7,3,1,6,2,5,4,8,9). 此即为贪婪算法的结果. 再对该个体的各个基因位的值做随机变动，或采用变异算子进行变异，可得到初始种群. 利用贪婪算法初始化种群，再通过最优保留的选择策略，保证遗传算法最终的匹配结果一定不差于贪婪算法.

3.3　适应度函数

匹配的目标在于求解相似度最大的匹配. 根据定义 11 中的节点匹配相似度定义，可以将适应度函数设为个体 m 中所有节点映射相似度之和，因此节点匹配的相似度适应度函数定义如下：

$$f = \sum_{i=1}^{m} s_{i,m(i)},$$

其中：$m(i)$ 为所求个体第 i 基因位上的值，$s_{i,m(i)}$ 为两流程节点相似度矩阵中第 i 行第 $m(i)$ 列的值. 求这些匹配上的节点的相似度之和，是最简单的一种适应度函数.

根据流程的结构相似度定义，可以分别得

$$\begin{cases} |s_n| = \sum_{i=1}^{n} x_i; \\ s_{nv} = \dfrac{|s_n|}{m+n}; \\ x_i = \begin{cases} 1, \text{if } s_{i,m(i)} > 0; \\ 0, \text{else}; \end{cases} \\ s_{ev} = \dfrac{|s_e|}{|E_1| + |E_2|}; \\ s_{bv} = \dfrac{2}{m + n - |s_n|} \sum_{i=1}^{n} x_i (1 - s_{i,m(i)}). \end{cases}$$

式中：$m(i)$ 为所求个体第 i 基因位上的值，$s_{i,m(i)}$ 为两流程节点相似度矩阵中第 i 行第 $m(i)$ 列的值. 进一步得到结构相似度的适应度函数：

$$f = 1 - F_{\text{avg}}(s_{nv}, s_{ev}, s_{bv}).$$

3.4　遗传算子设计

由于采用了非二进制编码，所以变异算子需要采用对应离散数字的变异方法. 在此采用映射互换模式的变异方法、基因段逆转、互换位置相结合的方法进行遗传处理以提高遗传多样性.

3.5　参数设定

种群规模根据问题的节点数进行设定. 例如，匹配中，节点较多的流程节点数为 n，则相应的种群规模可设为 $n \sim 2n$. 迭代代数为节点数 3 ~ 5 倍，或者采用某个估计函数 $Af(n)$，其中 A 为一个基准常数，$f(n)$ 为一个节点数的增函数. 交叉概率可设为 $P_c = 0.9$，变异概率为 $P_m = 0.01$.

最终的算法运行流程如下：

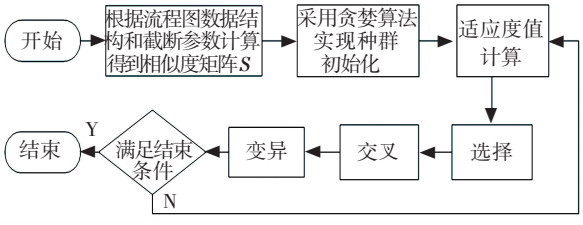


图 3　遗传匹配算法流程图

4　实　验

根据前述工作，该节根据流程节点数不同，将流程库中的流程分成 98 类，即 2~99 个节点的流程根据其节点数各成一类. 每一类流程一般有 8~12 个. 通过对不同类别流程的匹配计算，观察算法的性能表现. 在语义相似度的计算上采用了基于《知网》的语义知识库及其词汇语义相似度计算方法，并结合前文的分析实现流程配对和相似度计算. 实验时，首先通过确定截断值对测试结果的影响来确定合适的实验参数；其次，在确定的截断值条件下，通过匹配的相似度结果和匹配过程的时间消耗来确定匹配综合效果.

定义相似度时间成本函数 $C_{\text{cost}}(s,t)$：

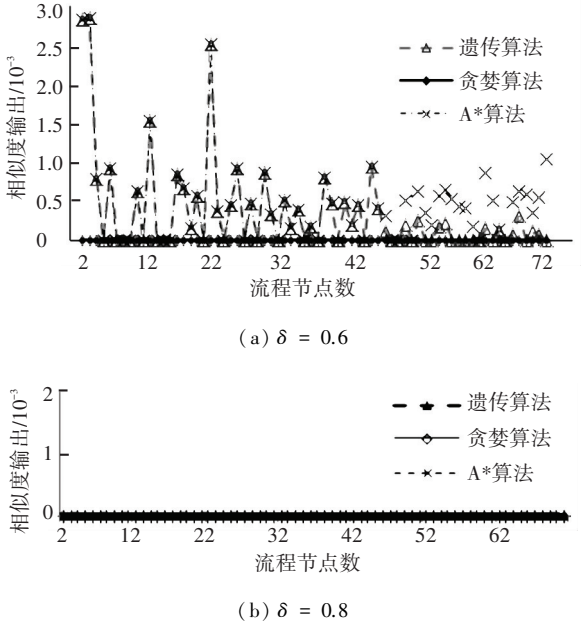
$$C_{\text{cost}}(s,t) = \frac{At(1-s)}{s - s_0 + \alpha} + C_n.$$

式中：t 为所耗费的时间，A 为比例调节常数，s 为求解的相似度值，s_0 为通过贪婪算法得到的相似度值，α 为一较小的常数，以避免分母为零；C_n 为两流程节点数的减函数. 通过函数 C_n 的调节，使得在节点增多时，函数成本适当减小. C_n 的确定与常数 A 和 α 相关，例如可取 $C_n = -A\mathrm{e}^{\alpha n}$. 根据实验数据特点，也可采用以下变形：

$$C_{\text{cost}}(s,t) = \ln\left(\frac{At(1-s)}{s - s_0 + \alpha} + C_n\right). \quad (1)$$

实验证明，截断值 (δ) 的选取对流程匹配算法的实用性具备重要影响，特别是 A * 算法在节点数逐渐增大约到 15 个节点时，平均耗时会开始超过 1 s/百对. 增加至 54 个节点时，程序耗时已经超过 2 d，因此采用文献[9]建议，δ 的下界取 0.6，此时得到的 A * 算法在相似度值和耗时两方面都处于可接受范围内. 实验中对贪婪算法、A * 算法、遗传算法进行包括时间复杂度、匹配结果以及时间成本代价等指标的对比，截断参数 δ 可选 0.6、0.8，当 $\delta = 0.6$

时,对比较果明显. 由于流程的相似度值差异值较小,为了清晰地表达实验结果,将 3 种算法得到的相似度值减去贪婪算法得到的值绘图,结果见图 4. 由图 4 可知,与 $\delta = 0.6$ 相比, δ 取 0.8 时,流程匹配相似度的绝对值整体下降了. 因为 δ 的变大,直接忽略相似度值较小的节点匹配,从而导致整体相似度值下降.



(a) $\delta = 0.6$

(b) $\delta = 0.8$

图 4 算法的匹配效果比较

从图 4 可以清楚地看到 3 种算法得到的相似度值的差异. 由于贪婪算法自身减掉自身的值得到的值始终是 0,因此,贪婪算法表现为一条恒为 0 的直线,延伸于坐标轴横轴方向上. 在 δ 为 0.6 时,可以看到遗传算法和 A * 算法大约在节点数达到 45 及其之前的匹配输出是完全重合的;当节点数>45 时,遗传算法的值开始处于 A * 算法之下. 这说明遗传算法在节点数超过 45 时,不能再保证相似度值是最优解. 当 δ 为 0.8 时,遗传算法和 A * 算法得到的相似度值与贪婪算法相同,减掉贪婪算法的值后结果都为一条恒为 0 的直线,延伸于横轴方向上,如图 4(b)所示. 因此, δ 超过 0.8 时,贪婪算法是最好的. 即当 δ 超过 0.8 时,截断值过于简化了算法. 然而如果实际需要,可以将 δ 设得比较大,如0.8. 但 δ 设得过大,可能使得匹配不到真正的最优解,从而导致匹配算法失去优势.

5 算法时间复杂度结果分析

在测试了截断值对匹配结果的影响后,实验取 $\delta = 0.6$ 对 3 种算法的时间复杂度进行测试. 图 5 为 3 种算法随着节点数增加,时间复杂度的情况. 横轴表示节点数,纵轴表示每 100 对流程匹配所耗费的时间. 由于时间跨度较大,图中纵轴采用时间对数坐标.

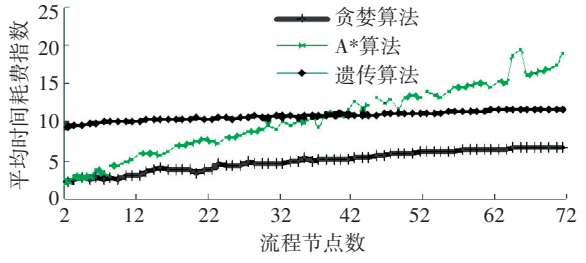


图 5 算法平均时间复杂度随节点增多变化趋势

由图 5 可知,贪婪算法在时间上表现最好. 同时,A * 算法在起初节点数较少时,时间和贪婪算法相当. 当节点数>5 时,A * 算法时间复杂度急剧增长,并在节点数为 37 附近,超越遗传算法呈剧烈上升趋势. 在节点数为 37 处,A * 算法的时间耗费约为 11(e^{11}ms 约为 12 s). A * 算法的时间复杂度基本保持随着节点数逐渐增长而增长,偶尔出现时间耗费降低或者突然增大的情况,说明 A * 算法的波动性较大. 遗传算法的时间复杂度基本稳定,始终保持在 11 附近并缓慢增长. 出现以上实验现象是因为 A * 算法受启发策略的影响很大,同时其复杂度随着节点数的增加会呈现指数增长趋势,最终总体上体现出快速增长并偶有波动. 而采用贪婪算法,随着节点数的提升,时间复杂度呈对数增长模式,因此能有效避免状态空间爆炸问题. 遗传算法的复杂度取决于其迭代次数和进化过程,具备复杂度小且可控的特性.

6 时间成本评比

基于式(1),参数采用了如下的取值: $A = 0.01$, $\alpha = 0.001$, $C_n = -A\mathrm{e}^{\alpha n}$. 对实验数据进行二次处理,得出时间成本效果对比曲线.

可以看到,图 6 的趋势与图 5 是一致的. 从图 6 可得贪婪算法的成本基本上一直维持在最低. 因为贪婪算法的时间复杂度很小,使得贪婪算法在一些应用场景下始终具有优势. 遗传算法在节点数<37 时,成本函数始终大于 A * 算法;超过 37 时,与 A * 算法相当,超过 43 个节点时,遗传算法相对于 A * 算法表现出优势,而且时间成本相对稳定,增长缓慢. 即当节点数超过 43 时,采用 A * 算法来求解匹配的最优值是不如遗传算法“划算”的. 综合前述实验结果,当流程节点达到一定数量时,采用遗传算法进行流程最优匹配是一个能兼顾匹配输出效果和时间成本的选择,对于实际应用来说,具备明显的折中优势.

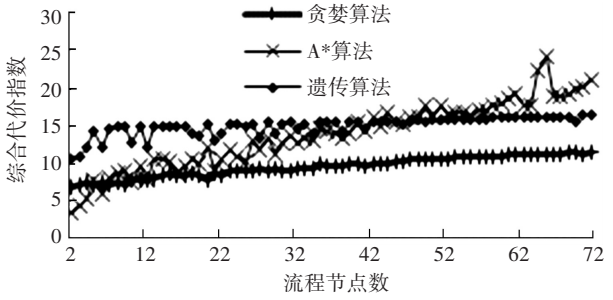


图 6　相似度计算综合代价随节点增多的变化趋势对比

7　结　论

1)从流程节点的语义相似度和流程结构匹配算法两方面进行了研究. 结合流程节点语义相似度和流程结构相似度的相关定义，导出了本文的流程相似度计算模型.

2)针对流程相似度寻优过程,设计了一种基于遗传算法的过程寻优算法.

3)对目标算法进行了实验验证,通过对实验数据的分析,证明了本文设计的流程匹配算法在匹配的时间复杂度和匹配输出方面的综合优化能力,对比已有的匹配过程寻优算法,本文算法体现出很好的实用价值.

鉴于时间和实验条件,本文虽然在流程匹配中尝试了对语义的考察,但还有进一步深入的空间,如实现本体语义的语义匹配,尝试其他新算法在匹配寻优过程中的应用等.

参考文献

[1] JIN T, WANG J, LA ROSA M, et al. Efficient querying of large process model repositories [J]. Computers in Industry,2013, 64(1): 41-49.

[2] WANG P, LIU H. Assessing sentence similarity using wordnet based word similarity [J]. Journal of Software, 2013, 8(6): 1451-1458.

[3] LI H, TIAN Y, CAI Q. Improvement of semantic similarity algorithm based on WordNet [C]// 2011 6th IEEE Conference on Industrial Electronics and Applications (ICIEA). Beijing: IEEE, 2011:564-567.

[4] 郭永利,卢颖颖. 基于 Lucene 对文件全文检索的研究与应用[J]. 微型电脑应用, 2014(1): 51-54.

[5] 李永春,丁华福. Lucene 的全文检索的研究与应用[J]. 计算机技术与发展,2010, 20(2): 12-15.

[6] 付永贵. 一种改进的余弦向量度量法文本检索模型[J]. 图书情报工作,2011(19): 115-119.

[7] BERRETTI S, DELBIMBO A, VICARIO E. Efficient matching and indexing of graph models in content-based retrieval[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2001, 23(10): 1089-1105.

[8] COLOMOPALACIOS R, GOMEZBERBIS J M, GARCIACRESPO A, et al. SeMatching: using semantics to perform pair matching in mentoring processes [C]//2nd World Summit on the Knowledge Society. Chania: Springer Verlag, 2009:137-146.

[9] DIJKMAN R, DUMAS M, GARCIABANUELOS L. Graph matching algorithms for business process model similarity search [C]//7th International Conference on Business Process Management. Ulm: Springer Verlag, 2009:48-63.

[10] THANUJA M K, MALA C. A search tool using genetic algorithm [M]//Information Technology and Mobile Communication. Chania: Springer, 2011: 138-143.

[11] HONDA K, NAGATA Y, ONO I. A parallel genetic algorithm with edge assembly crossover for 100,000-city scale TSPs [C]// 2013 IEEE Congress on Evolutionary Computation (CEC). Cancun: IEEE,2013:1278-1285.

[12] CEKMEZ U, OZSIGINAN M, SAHINGOZ O K. Adapting the GA approach to solve Traveling Salesman Problems on CUDA architecture [C]// 2013 IEEE 14th International Symposium on Computational Intelligence and Informatics (CINTI). Budapest: IEEE, 2013:423-428.

[13] 葛斌,李芳芳,郭丝路,等. 基于知网的词汇语义相似度计算方法研究[J]. 计算机应用研究, 2010(9): 3329-3333.

[14] 周生宝,郭俊芳. 本体映射中概念相似度计算的改进[J]. 山西大同大学学报(自然科学版), 2008(4): 38-40.

[15] 王婷. 本体相似度研究[J], 电脑知识与技术(学术交流), 2007(6): 1609-1611.

[16] 张艳霞,张英俊,潘理虎,等. 一种改进的概念语义相似度计算方法[J]. 计算机工程, 2012(12): 176-178.

[17] 胡哲,郑诚. 改进的概念语义相似度计算[J]. 计算机工程与设计,2010(5): 1121-1124.

[18] DIJKMAN R, DUMAS M, DONGEN B V, et al. Similarity of business process models: Metrics and evaluation [J]. Information Systems,2011, 36(2): 498-516.

[19] ZHU J, PUNG H K. Process matching: A structural approach for business process search [C]//Computation World: Future Computing, Service Computation, Adaptive, Content, Cognitive, Patterns, Computation World 2009. Athens: IEEE Computer Society, 2009:227-232.

(编辑　杨　波)