计算机科学 1995 Vol. 22№. 5

并行算法可伸缩性的E微商分析法*)

林 洪 陈国良
(中国科学技术大学计算机系 合肥 230027)

摘 要 The E-differentiation method, which is derived from the idea of isoefficiency analysis and used for analysing scalability of parallel algorithms on parallel architectures, is proposed in this paper. E-differentiation can overcome the incompletion of other metrics for analyzing scalability which confine the manner of increasing of W w.r.t. p under some condition; and it is easy to give the equivalent expression of these metrics in terms of E-differentiation thereof, relationships between these metrics become more definitive. We conclude with an example of scalability analysis using E-differentiation and compare it with isoefficiency function method.

关键词 MPP(Massively Parallel Processing), Parallel algorithm, Scalability analysis, E-differentiation.

1. 引 言

并行算法在并行体系结构上的可伸缩性分析(Scalability analysis)是目前巨量并行处理MPP研究的中心问题之一。可伸缩性作为巨量并行机上并行算法的主要性能指标，揭示了在高性能计算机HPC上求解巨复杂(Grand Challenge)问题的有效性。

所谓并行算法的可伸缩性是指并行算法在可伸缩的并行体系结构上有效地利用不断增加的处理机的能力[1]。目前已有至少十来种衡量并行算法可伸缩性的标准(Metric)，如等效率函数[2]、伸缩加速[3]、串行比例(Serial fraction)f[4]、开销函数 $\Phi(p)$[5]、数据效率函数DEF及处理机效率函数PEF[6]等。不同的标准从不同的角度定义了并行算法在特定体系结构上的可伸缩性，因此找出一个最佳标准或用一个标准取代另一个标准都不现实[7]。笔者认为并行算法的可伸缩性的物理定义是明确的，形式定义的困难在于无法简明地给出影响并行算法可伸缩性的基本因素及这些基本因素间关系的形式表达，所以下一步研究的关键在于明确各种影响可伸缩性的基本因素以及各种可伸缩性标准的内在关系。

现有可伸缩性分析方法的一个共同特征是，以某种方式限定问题尺寸(或工作量)W随处理机数p的增长而增长的方式，根据某个量的变化情况，作出并行算法可伸缩性的度量。例如，等效率分析法限定E=const，考察W随p增长的函数 $W=f_E(p)$，以 $f_E(p)$的增长速率作为算法可伸缩性的度量[2]；伸缩加速分析法限定W随p线性增长，以加速比S接近线性的程度作为算法可伸缩性的度量[3]；串行比例分析法令W=const，以p增长时等效串行比例f的增长速率作为算法可伸缩性的度量，其中f定义为 $f=(1/S-1/p)/(1-1/p)$，S为加速比，当通讯等开销为零时，f即为串行比例；而当通讯等开销不为零时，f还包含这些开销与有效工作量的比值[4]。开销函数分析法令W的增长速率充分大于p的增长速率，以开销函数 $\Phi(p)=T_p/C(W_s+W_p/p)$的增长速率作为算法可伸缩性的度量，其中，T_p为p个处理机执行程序的时间，W_s、W_p分别为程序的串行和并行分量，C为常数[5]。处理机效率函数法限定W=const，以满足 $T_p=O(W/p)$的p的上限值为算法可伸缩性的度量；数据效率函数法则限定p=const，以满足同式的W的下限值作为算法可伸缩性的度量[6]。等等。对于同一个系统结构上两个并行算法的可伸缩性的优

*)本文由863计划资助。林 洪 博士研究生，研究方向为并行处理。陈国良 教授，系主任，博士生导师，主要研究领域是非数值计算的并行算法、并行/分布式计算、神经网络和智能计算理论等。

劣，不同的分析方法有可能得出不同的结论，而且其间的相互关系也极不明确。事实上，算法的可伸缩性取决于 p 的改变对算法性能的影响和 W 的改变对算法性能的影响的综合效果。因此，限定 W 随 p 的增长方式必然不能全面考察 W 对性能的影响和 p 对性能的影响的相互关系。所以，分析算法的可伸缩性，应该把 W、p 作为自变量，在 W-p 平面上考察算法性能。本文参照了等效率分析的思想，即以效率分析作为分析算法性能的手段，进而提出(E，W，p)空间上的 E 微商法。

2. E 微商法

2.1 等效率分析

对于大多数并行算法而言，当 p 增加时，由于 Amdahl 定律的制约，不增加问题规模，加速就不会线性增长[7]，因而效率要下降；但如果问题规模随 p 按一定比例增长，可以得到线性加速，$S=\alpha\cdot p$(α 为常数且 $\alpha\leqslant 1$)，因而效率 $E=S/p=\alpha$ 为常数。Kumar 等人从这一点出发，定义了等效率函数。若 T_e 表示有效计算时间(即算法的串行计算时间，或记为 T_1，它与工作量 W 的关系为 $T_e=t_c\cdot W$，其中 t_c 为一个操作步的计算时间)；以 T_o 表示在 p 个处理机上并行执行时的额外开销(包括通讯开销、同步开销、等待时间等)，则有：$T_p=(T_e+T_o)/p$，加速比 $S=T_e/T_p=p/(1+T_o/T_e)$，效率 $E=1/(1+T_o/T_e)$，易见 $E=\alpha$ 的充要条件是：$T_e/T_o=k$(k 为常数且有 $\alpha=k/(1+k)$)，即处理机数 p 增加时，需保持有效计算时间 T_e 和开销时间 T_o 成比例。如果 T_o 随 p 的增加而增加，则必须增加 W(相应地增加 T_e)，而且由于增加 W 所引起的 T_o 的增加速度低于 W 的增加速度，才能维持 $T_e/T_o=k$，对于一大类并行算法，这一条件是满足的。这时 W 与 p 的对应关系就是等效率函数，记为 $W=f_E(p)$。若 $f_E(p)$是 p 的线性函数，则算法是最佳可伸缩的(Perfectly scalable)；$f_E(p)$随 p 增长越快，算法的可伸缩性越差。如图 1 所示：

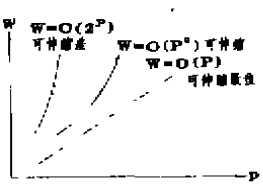

图 1 算法的等效率曲线

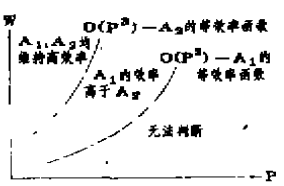

图 2 两个算法的效率比较

可以证明，当算法的串行比例不为零时，$f_E(p)$不会低于线性。事实上，如果算法的串行分量为 W_s，则等待时间为 $t_c\cdot W_s\cdot(p-1)$，记 T'_o 为除去等待时间之外的开销时间，则有：$T_o=t_c\cdot W_s\cdot(p-1)+T'_o$，现令 $T_e=kT_o$，则有：$W=k\cdot W_s\cdot(p-1)+(k/t_c)\cdot T'_o$，由于 $W_s\geqslant 1$，故 $W\geqslant O(p)$。

等效率分析对一大类并行算法被证明是行之有效的，其主要优点是用一个表达式简明地刻画了一个并行算法在一个并行体系结构上的性能特征[1]。

然而某些算法的等效率函数可能不存在，可以证明，等效率函数存在且单调增加的必要条件是：若 W 不变而 p 增加时，E 单调减小，则当 p 不变而 W 增加时，E 必单调增加。但该条件不是充分的，事实上，若算法的串行比例 $f=W_s/W$ 为常数，$T'_o=O(p\cdot\sqrt{W})$，则：

$$E=1/(1+f(p-1)+\frac{T'_o}{T_e})$$

$$=1/(1+f(p-1)+c\cdot\frac{p}{\sqrt{W}})(c\text{ 为常数})$$

显然该算法满足上述条件，但易验证其等效率函数不存在。由此可见等效率限定是苛刻的。与此相对照，伸缩加速、串行比例、开销函数等方法的适用性则较好，对任何算法，总能给出其可伸缩性判定。

此外，等效率函数描述了为维持某个效率不变，W 随 p 的增长速度，但并没有说明 W 随 p 的变化不遵循等效率曲线时效率 E 的改变情况。例如，如图 2 所示，算法 A_1 和 A_2 的等效率函数分别为 $O(p^2)$和 $O(p^3)$，可以判断 A_1 的可伸缩性优于 A_2。假设 W 的增长速度超过 $O(p^2)$，则 A_1、A_2 均可得到近于线性的加速；如果 W 的增长速度介于 $O(p^2)$和 $O(p^3)$之间，则 A_1 的加速必然高于 A_2；但如果 W 的增长速度低于 $O(p^2)$，则 A_1、A_2 的加速就高就低就无法判断，A_1 的加速可能高于 A_2，但也可能 A_2 的加速超过 A_1，如果 A_2 的效率对 W 的改变的敏感性低于 A_1 的话。

可见，E 对 W 的变化的敏感性不能由等效率函数表明。Kumar 认为该问题的解决需引进一补充标准以说明 W 的增长速率低于等效率函数时 E 下降的速度[8]。这样，等效率分析至少需要两个辅助标准：一个用于等效率函数不存在的算法的判定，Kumar 认为开销函数 $\Phi(p)$较佳，尽管 $\Phi(p)$也并非充分[1]；另一个用于说明 W 以异于等效率曲线的方式变化时 E 的变化趋势。事实上，等效率函数的这一缺点正是等效率限定条件造成的，因为 W 成为 p 的函

数，自变量只有一个 p，必然丢失 W 对 E 的影响的信息。所以，解决问题的方法不一定在于增加辅助标准，可能在于放弃等效率限定条件。下面，试提出一种可伸缩性分析的一般方法——E 微商法。

2.2 E 微商法

在(E,W,p)空间上 E 对 W 的变化率定义为 $\frac{\Delta E}{\Delta_{min}W}$，$\Delta_{min}W$ 为算法的问题尺寸的最小改变量，注意对不同的算法或同一个算法的不同 W，$\Delta_{min}W$ 可能不相同，如快速排序算法的 $\Delta_{min}W$ 为 $(n+1)\log(n+1)-n\log n$（n 为输入尺寸）；E 对 p 的变化率定义为 $\frac{\Delta E}{\Delta_{min}p}$，$\Delta_{min}p$ 为并行体系结构上 p 的最小改变量，对不同的结构或同一结构的不同 p，$\Delta_{min}p$ 也可能不同，如 mesh 的 $\Delta_{min}p$ 为 $(m+1)^2-m^2=2m+1$（$p=m^2$）。所以，(E,W,p)空间上并不是所有点(W,p)上都有 E 值，但我们可以用连续可微曲面 $E=E(W,p)$ 来拟合 W-p 平面上的 E 分布值，使 E 对 W 和对 p 的变化率均等于相应点上 E 对 W 和对 p 的偏微商，即 $\frac{\Delta E}{\Delta_{min}W}=E'_W$，$\frac{\Delta E}{\Delta_{min}p}=E'_p$，并使 E'_W 和 E'_p 在相邻 E 分布点区间上单调连续。显然用 E'_W 和 E'_p 代替 $\frac{\Delta E}{\Delta_{min}W}$ 和 $\frac{\Delta E}{\Delta_{min}p}$ 不改变对算法性能的分析。

所谓并行算法的可伸缩性，就是在任一点(W,p)上，E 值是否能保持较高的程度（或者存在 W_0，p_0，当 $W>W_0$ 或 $p>p_0$ 时，E 值能否保持较高。为简便起见，下面对这种情况不加说明）。

定义 1 算法 A_1 的绝对可伸缩性优于 A_2，iff 对 W-p 平面上任意点(W,p)，均有 $E_1(W,p)\geqslant E_2(W,p)$，其中 $E_1(W,p)$、$E_2(W,p)$ 分别是 A_1、A_2 的 E 分布函数。

得到算法的精确 E 分布函数通常是困难的，而且也没有必要，因为对于任意性能指标，我都只关心它们的线性主项，而忽略次项。也就是说，我们关心的是 E 随 W、p 的增长而变化的性态。在这种意义下，算法的可伸缩性是指当 W、p 增长时 E 的变化率。

定理 1 在过 W-p 平面上一点(W,p)的任一曲线沿 W、p 增加方向上算法 A_1 的 E 增长率大于算法 A_2 的 E 增长率的充要条件是：

$$E'_{1W}\geqslant E'_{2W} \quad 且 \quad E'_{1p}\geqslant E'_{2p}$$

其中 E_1、E_2 的含义与定义 1 中相同（以下不再说明）。

证明：因为 $dE_1=E'_{1W}dW+E'_{1p}dp$，$dE_2=E'_{2W}dW+E'_{2p}dp$，则 $dE_1\geqslant dE_2$ 当且仅当 $E'_{1W}dW+E'_{1p}dp\geqslant E'_{2W}dW+E'_{2p}dp$，此式在 $dW\geqslant 0$ 且 $dp\geqslant 0$ 时成立，当且仅当 $E'_{1W}\geqslant E'_{2W}$ 且 $E'_{1p}\geqslant E'_{2p}$。 □

如果在任一点(W,p)的任一方向上 A_1 的 E 增长率均大等于 A_2，则认为 A_1 的可伸缩性优于 A_2，根据定理 1，则有：

定义 2 算法 A_1 的相对可伸缩性优于 A_2，iff 在 W-p 平面上任一点(W,p)上均有：

$$E'_{1W}\geqslant E'_{2W} \quad 且 \quad E'_{1p}\geqslant E'_{2p}$$

相对可伸缩性包含两重含义，即当固定 W 时，增加 p 而 E 降低较慢，而当固定 p 时，增加 W 而 E 增加较快。如果我们把 $E'_{1W}\geqslant E'_{2W}$ 作为 A_1 的固定 p 可伸缩性优于 A_2 的判据；把 $E'_{1p}\geqslant E'_{2p}$ 作为 A_1 的固定 W 可伸缩性优于 A_2 的判据，则 A_1 的相对可伸缩性优于 A_2 当且仅当 A_1 的固定 p 可伸缩性和固定 W 可伸缩性均优于 A_2。

固定 p 可伸缩性判定和固定 W 可伸缩性判定固然很弱，不能作为可伸缩性判断标准，而相对可伸缩性判定却太强，因为它蕴含了等效率函数判定。

定理 2 若在相对可伸缩性判定下 A_1 的可伸缩性优于 A_2，则在等效率函数判定下 A_1 的可伸缩性优于 A_2（如果等效率函数存在的话）。

证明：因为 $E=\text{const}$ 当且仅当 $dE=0$ 即 $E'_W\cdot dW+E'_p\cdot dp=0$ 当且仅当 $\frac{dW}{dp}=-\frac{E'_p}{E'_W}$（注意等效率函数存在的必要条件是 $E'_p<0$ 且 $E'_W>0$）。又因为 A_1 的相对可伸缩性优于 A_2，即 $E'_{1W}\geqslant E'_{2W}$ 且 $E'_{1p}\geqslant E'_{2p}$，亦即 $-\frac{E'_{1p}}{E'_{1W}}\leqslant -\frac{E'_{2p}}{E'_{2W}}$，所以 $\frac{dW^{(A_1)}}{dp}\leqslant\frac{dW^{(A_2)}}{dp}$，即 A_1 的等效率函数的增长速率小等于 A_2 的等效率函数的增长速率，即在等效率函数判定下 A_1 的可伸缩性优于 A_2。 □

注意：至此我们在某一点(W,p)上比较两个算法 A_1、A_2 的性能时，并未考虑 $E_1(W,p)$ 是否等于或近似于 $E_2(W,p)$。如果要求在同等 E 值下判定可伸缩性，则要对任一 p 值，取 W_1、W_2，使 $E_1(W_1,p)\sim E_2(W_2,p)$，在 (W_1,p) 和 (W_2,p) 上分别计算 A_1、A_2 的可伸缩性。

如前所述，等效率分析中的等效率限定条件是很强的，它排除了一大类算法的等效率函数的存在性，比如串行比例为常数的算法。但在任一点(W,p)

上$-\frac{E'_p}{E'_w}$总是存在的，因此可作为算法可伸缩性的判断标准。

定义 3 算法 A_1 的保效率可伸缩性优于 A_2，iff 在 W-p 平面上任一点(W,p)均有：

$$-\frac{E'_{1p}}{E'_{1w}} \leqslant -\frac{E'_{2p}}{E'_{2w}}$$

参考定理 2 的证明过程不难得到：

定理 3 若算法 A_1 的相对可伸缩性优于 A_2，则 A_1 的保效率可伸缩性优于 A_2。

定理 4 若算法 A_1、A_2 的等效率函数存在且在等效率分析下 A_1 的可伸缩性优于 A_2，则 A_1 的保效率可伸缩性优于 A_2。

注意定理 3 和定理 4 的逆定理均不成立，这说明保效率可伸缩性判定较弱，它甚至不能推出固定 W 可伸缩性判定和固定 p 可伸缩性判定。保效率可伸缩性的物理意义是：因增加 W 而带来的 E 的增加能够补偿因增加 p 而带来的 E 的减少的"能力"越大，则算法的可伸缩性越好。可见，保效率可伸缩性判定非常符合可伸缩性的物理定义，其表达的简明性也较好。由于放弃了等效率限定条件，对任何并行算法，均可作出可伸缩性判定，其适用性优于等效率分析法。

引入了 E 微商，E 随 W 的变化率问题也迎刃而解了。事实上，该问题等价于固定 p 可伸缩性判定问题的逆问题，即对算法 A_1、A_2，当 W 随 p 的增加而增加的速度低于两者的等效率函数时，当且仅当 A_1 的固定 p 可伸缩性优于 A_2，则 A_1 的效率降低速度高于 A_2，从而 A_1 的加速更低于线性。

2.3 E 微商法与其他标准的关系

限于篇幅，这里只给出伸缩加速和串行比例在 E 微商法中的等效表达。定理的证明一律略去。

伸缩加速是指当 $W=m\cdot p$ 时(m 为常数)，若 p 增加时的加速比 S 越接近线性，则算法的可伸缩性越好[3]。

定理 5 算法 A_1 在伸缩加速判定下的可伸缩性优于 A_2，当且仅当在直线 $W=m\cdot p$(m 为常数)上，有：$E_1+p(E'_{1p}+m\cdot E'_{1w})\geqslant E_2+p(E'_{2p}+m\cdot E'_{2w})$成立。

还有两种伸缩加速，一种是存储受限伸缩(Memory constrained scaling)[9]，即当 p 增加时，系统相应的存储容量按比例增长 $M=k\cdot p$(k 为常数)。可处理的问题规模上限也随之增长，$S(n)=M=k\cdot p$(S(n)为算法空间复杂度，n 为输入尺寸)。

定理 6 对于算法 A，若 $S(n)=l\cdot n$(l 为常数)，则 A 在存储受限伸缩加速判定下可伸缩性越好，当且仅当在曲线 $W=W(n)$上($n=c\cdot p, c=k/l$ 为常数)，$E+p(E'_p+c\cdot E'_w\cdot W'_n)$越大。

另一种是时间受限伸缩(Time constrained scaling)[10]，即当 $T_p=t$(t 为常数)时的加速 S。

定理 7 算法 A 在时间受限伸缩加速判定下可伸缩性越好，当且仅当在曲线 $W=c\cdot p\cdot E$($c=t/t_c$ 为常数)上，$(E+pE'_p)/(1-c\cdot p\cdot E'_w)$越大。

对照以上几种伸缩方式的伸缩加速在 E 微商法中的等效判据，可以看到它们的判定结果可能互不相同，甚至可以得出完全相反的结论。原因在于当 W 随 p 的增长而变化的方式不同时，其效率的变化性态可能完全不同，因此从一般性考虑，笔者认为这些标准不及等效率函数。

定理 8 算法 A 在串行比例判定下可伸缩性越好，当且仅当 $W=c$(c 为常数)时，$E+pE'_p$越大。

3. 用 E 微商法分析算法的可伸缩性

所有点对的最短路径(All-pairs shortest path)并行算法是被研究得较多的典型并行算法，其中有一大类来源于 Floyd 串行算法，它以一个三重循环为主体，故 $W=O(n^3)$。下面在 mesh 结构上讨论两个典型 Floyd 并行算法：网格(Checkerboard)算法和流水线网格算法(Pipelined checkerboard)的可伸缩性。

在 mesh 结构上，网格算法的每一重循环内要进行 $O(n^2/p)$步计算和 $O(n)$步通讯[8]，于是其效率 $E_1=1/(1+T_0/T_e)=1/(1+\frac{n}{n^2/p})=1/(1+\frac{p}{n})$，因 $W=O(n^3)$，故 $E_1=1/(1+p/W_1^{\frac{1}{3}})$，可算得 $-\frac{E'_{1p}}{E'_{1w_1}}=\frac{W_1}{p}$。计算中略去了常数因子，以下类同。

同样在 mesh 上，流水线网格算法的总通讯开销为 $O(np)$[8]，于是，$E_2=1/(1+\frac{np}{n^3})=1/(1+\frac{p}{W_2^{1/3}})$，可得 $-\frac{E'_{2p}}{E'_{2w}}=\frac{W_2}{p}$。注意对任意 p，必须比较 $E_1\sim E_2$ 的点上的保效率可伸缩性，所以 W_1 与 W_2 未必相同。事实上，令 $E_1\sim E_2$，有 $W_1=W_2^3$，于是可得 $-\frac{E'_{1p}}{E'_{1w_1}}>-\frac{E'_{2p}}{E'_{2w_2}}$，所以在保效率可伸缩性判定标准下，网格算法的可伸缩性较流水线网格算法差。

若采用等效率分析，对网格算法，若 $E_1=$const，

须有 $O(n^2/p)\sim O(n)$，即 $n\sim O(p)$，于是等效率函数为 $W_1=O(p^3)$；对流水线网格算法，若 $E_2=const$，须有 $O(n^3)\sim O(np)$，即 $n\sim O(p^{1/2})$，等效率函数为 $W_2=O(p^{3/2})$。所以在 mesh 上，网格算法的可伸缩性较流水线网格算法差[5]。这与 E 微商法的判定结果一致。

现在研究当 W 随 p 的增长速率小于 $O(p^{3/2})$ 时（即小于两个算法的等效率函数），哪个算法的效率降低速度更快。可算得 $E'^{*}_{1W}=p/W^{2/3}(p+W^{1/3})^2$，$E'_{2W}=p/W^{1/3}(p+W^{2/3})^2$，比较两式可得：当 $W\leqslant p^2$ 时 $E'_{1W}\leqslant E'_{2W}$，而当 $W>p^2$ 时 $E'_{1W}>E'_{2W}$。现在 $W<O(p^{3/2})<O(p^2)$ 所以 $E'_{1W}\leqslant E'_{2W}$，即网格算法的效率降低速度比流水线网格算法要慢，亦即网格算法在 $W<O(p^{3/2})$ 时更易接近线性加速。

4. 结论

以效率作为算法可伸缩性研究中的性能指标是等效率分析法的微妙之处，符合可伸缩性的物理含义，但因等效率限定条件制约了 W 随 p 变化而变化的方式，造成了不能全面反映性能在 W-p 空间上的变化特征，也限制了其适用范围。笔者从 W、p 均作为自变量以全面把握 E 在 W-p 平面上的变化趋向出发，提出了 E 微商法。分析表明，E 微商法由于其一般化，能够解决等效率分析法未能解决的一些问题。它对算法的可伸缩性的判定结果和等效率分析法一致，但适用于所有并行算法；它可以在 W 随 p 变化而变化的方式不遵循等效率函数时，作出性能分析。此外，用 E 微商法可以等效地表达其他可伸缩性判断标准，使它们更易于比较。进一步的研究有望解决算法可伸缩性的形式定义问题。

致谢 笔者和本系博士研究生顾乃杰等同志，及 93 级硕士研究生李晓峰、江松等同学一同进行了并行算法可伸缩性的讨论，受到不少启发。文中等效率函数不低于线性的结论，首先由顾乃杰作出。在此一并致谢。

参考文献

[1]Kumar, V. et al., Analyzing Scalability of Parallel Algorithms and Architectures. Technical report, Army High Performance Computing Research Center, University of Minesota, 1992

[2]Kumar, V., Rao, V. N., Parallel Depth-First Search, Part I: Analysis. Intl. J. of Parallel Programming, 1987

[3]Gustafson, J. L. et al., Development of Parallel Methods for a 1024-processor Hypercube. SIAM J. on Scientific and Statistical Computing. 9(4), 1988

（下转第 74 页）

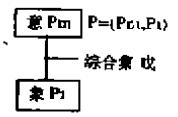

图 3

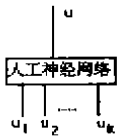

图 4

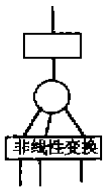

图 5

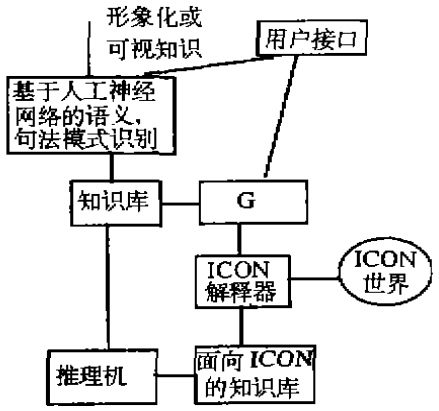

图 6 具有可视特性的知识库系统的一般结构

如果一个模式 P(x,u)由 k 个子模式 $p_1,\cdots p_k$ 构成,p_i 的属性为 u_i,i=1,2,…,k。所谓语义函数,就是如何由 $u_1,u_2,\cdots,u_k$ 而得到它的属性 u。语义函数是一个非线性的映射,我们可以用人工神经网络来构成,如图 4(语义函数的构建原理)、图 5(语义函数构成)所示。而这种人工神经网络的参数即"权"值,是通过有教师的学习(Supervised Learning)来完成的,由教师把握住宏观的情况,所以人在这一过程中起到积极的作用,这样一来,确定语义函数就有了一定的办法。

通过以上讨论,我们可给出具有可视特性的知识库系统的一般结构(图 6)。

作者已根据本文的思想,在近似推理系统中进行了可视化的尝试—动态显示近似推理系统的求解过程[6,7]。然而,在知识库系统可视化方面,需研究解决的问题还很多,这些都有待进一步探讨。

参考文献

[1] A. E. Kanfman, Visualization, Computer, Vol. 27, No. 7, 1994

[2] S. K. Chang, Principles of Pictorial Information Systems Design, Prentice-Hall, Inc., New Jersey, 1989

[3] S. K. Chang, ICON Semantics-A Formal Approach to ICON System Design, IJPRAI, Vol. 1, No. 1, 1987

[4] S. K. Chang et al, The SIL-ICON Compiler-An ICON-Oriented System Generator, IJPRAI, Vol. 2, 1988

[5] 戴汝为,基于人工神经网络的语义、句法模式识别,见《智能计算机基础研究》,北京清华大学出版社,1994

[6] Z. L. Zhang(张自力), An Approximate Reasoning System: Design and Implementation, Int. J. of Approximate Reasoning, Vol. 9, 1993

[7] Z. L. Zhang (张自力), Visualization of Approximate Reasoning Systems, J. of Southwest China Normal University (Natural Science Series), No. 4, 1995

(上接第 5 页)

[4]Karp, A. H. et al., Measuring Parallel Processor Performance. CACM, 33(5), 1990

[5]Zorbas, J. R. et al., Measuring the Scalability of Parallel Computer Systems, Proc. of Supercomputing '89, 1989

[6]Chandran, Davis, An Approach to Parallel Vision Algorithms. In Porth. R., editor, Parallel Processing, SIAM, 1987

[7]Amdahl, G. M., Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities. In AFIPS Conf. Proc., 1967

[8]Kumar, V., Singh, V., Scalability of Parallel Algorithms for the All-Pairs Shortest Path Problem: A Summary of Results. Proc. of 1990 Intl. Conf. on Parallel Processing, 1990

[9]Gustafson, J. L., Reevaluating Amdahl's Law. CACM, 31(5), 1988

[10]Worley, P. H., The Effect of Time Constraints on Scaled Speedup. SIAM J. on Scientific and Statistical Computing, 11(5), 1990