基于 prüfer 数的遗传算法求解度约束最小树问题

牧云志,周根贵
MU Yun-zhi,ZHOU Gen-gui

浙江工业大学 经贸管理学院,杭州 310023
College of Business Administration,Zhejiang University of Technology,Hangzhou 310023,China
E-mail:muriel2001@163.com

MU Yun-zhi,ZHOU Gen-gui.Genetic algorithm based on prüfer number for solving Degree-Constrained Minimum Spanning Tree Problem.Computer Engineering and Applications,2008,44(12):53-56.

Abstract: The Degree-Constrained Minimum Spanning Tree problem (DCMST) is difficult to be solved because of its NP-hard complexity.But it's very important because of its value in practice.In this paper,we discuss how to solve this problem by means of genetic algorithm based on prüfer number.We present it by using C and MATLAB programs.The numerical analysis shows the effectiveness of the genetic algorithm in practice.
Key words: prüfer number;Genetic Algorithm(GA);Minimum Spanning Tree(MST);degree-constrained

摘 要:度约束最小树问题属于 NP-完全问题,是一类比较难解的问题,但在现实中具有非常重要的应用价值。探讨了如何将基于 prüfer 数的遗传算法应用于该问题,并给出了相应的算法。采用 C 语言和 MATLAB 的混合编程实现该算法,数值分析的结果显示了遗传算法求解该问题的有效性及其应用价值。
关键词:prüfer 数;遗传算法;最小生成树;度约束
文章编号:1002-8331(2008)12-0053-04 **文献标识码:**A **中图分类号:**TP301

1 引言

生成树在大多数网络设计和分析问题中扮演着重要的角色,已被普遍认为是通信网络中一种最基本的拓扑结构。最小生成树(Minimum Spanning Tree,MST)是通信网络设计的最佳拓扑结构。该问题在组合优化中历史悠久,是 Boruvka 于 1926 年提出的,目的是寻找电力线网络最优的布局[1]。在实际问题中,生成树的结构往往需要满足各种不同的约束条件,于是产生了一系列带约束的最小树问题。例如,在最小生成树问题中,对连接到每个节点的边数加上一定的限制,即各节点的度(degree)受到一定的限制,各节点度值不超过给定的数值。在现实世界中,有许多这样的例子,比如通信网络中为了防止节点故障带来的脆弱性,对节点的度也有一定的限制。这种带有节点度约束的最小生成树问题被称之为度约束最小生成树(Degree-Constrained Minimum Spanning Tree,DCMST)问题[1]。Narula 和 Ho,Volgenant 等提出了解决这个问题的启发式算法[2,3],马良和蒋馥等提出一种快速近似算法求解该问题[4,5],近几年来,许多学者运用遗传算法(GA)研究 DCMST 问题[6-8]。

在优化问题中,GA 作为一种新的现代启发式算法,已显示了非常广泛的应用前景,并已应用到最优控制、运输问题、排序问题、生产计划、资源分配、统计及模式识别等领域。对于 DCMST 问题,本文采用端点编码法中的 prüfer 数来给树编码,因为 prüfer 数编码包含节点的度的信息,在 prüfer 数编码中度为 d 的端点正好出现 $d-1$ 次[1],并能保证在进行遗传操作时,始终保持遗传编码代表一棵最小树,这样就可以有效地求解 DCMST 问题,另外,该编码法需要 $O(n\log n)$ 的计算时间和 $O(n)$ 的空间[8]。本文将运用基于 prüfer 数编码的遗传算法求解 DCMST 问题,并与启发式算法、快速近似算法的结果进行比较研究。

2 DCMST 问题的数学模型

考虑赋权的无向连通图 $G=(V,E,\boldsymbol{W})$,即一个网络。其中 $V=\{v_1,v_2,\cdots,v_n\}$ 是节点的有限集合,$n=|V|$,$E=\{(i,j)|i,j\in V\}$ 是边的有限集合,$m=|E|$,$\boldsymbol{W}=[w_{ij}]_{n\times n}$ 为权矩阵,用来表示相邻节点间的距离或费用,$w_{ij}=w_{ji}$,$w_{ii}=+\infty$。如果图 G 的一个子图 G' 是一棵包含 G 中所有节点的树,则 G' 为 G 的生成树。MST 问题就是在 G 的所有生成树中寻找具有最小权重的子图。

设 $x_{ij}(i,j=1,2,\cdots,n)$ 是 0-1 型决策变量,$x_{ij}=1$ 表示节点 i 与节点 j 之间有线路连接,否则无线路连接。设各节点的度约束为 $b_i(i=1,2,\cdots,n)$,特别当 $b_i\geqslant n-1$ 时,即为无约束最小树问题,当 $b_i=2$ 时,即为货郎担问题(TSP),就一般情况而言,DCMST 是一类 NP-完全问题[4]。DCMST 问题的数学模型可以用如下的整数规划表示:

基金项目:国家自然科学基金(the National Natural Science Foundation of China under Grant No.70671095)。
作者简介:牧云志(1983-),女,硕士研究生,主要研究领域为复杂网络拓扑结构设计与优化;周根贵(1958-),男,博士生导师,教授,主要研究领域为网络优化、遗传算法与软计算等。
收稿日期:2007-08-17 **修回日期:**2007-11-19

$$\min C(x)=\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} w_{ij}x_{ij} \tag{1}$$

$$\text{s.t.}\begin{cases}\sum\limits_{i=1}^{n-1}\sum\limits_{j=i+1}^{n} x_{ij}=n-1 & (2)\\ \sum\limits_{i\in S}^{n-1}\sum\limits_{\substack{j\in S\\ j>1}}^{n} x_{ij}\leqslant |S|-1,\ S\subseteq V\backslash\{1\},\ |S|\geqslant 2 & (3)\\ 1\leqslant \sum\limits_{j=1}^{n} x_{ij}\leqslant b_i,\ i=1,2,\cdots,n & (4)\\ x_{ij}\in\{0,1\},\ i,j=1,2,\cdots,n & (5)\end{cases}$$

一棵 n 个节点的树必有 $n-1$ 条边，约束条件（2）是生成树恒为真的必要条件；约束条件（3）表示无环；约束条件（4）为度约束并保证图 G 的所有节点都包含在树中。

3　遗传算法及其实现

3.1　编码与解码

编码过程：将一棵树转化为一个 prüfer 数。

步骤 1　令 i 为树 T 中标号最小的叶子节点，j 为其关联节点。则 j 为 prüfer 数编码 P 的第一个数字，编码顺序从左到右。

步骤 2　删除树 T 中的节点 i 及边 (i,j)。

步骤 3　重复上述步骤，直到剩下一条边。得到一个 $n-2$ 个介于 1 和 n 之间的数字排列。

解码过程：将一个 prüfer 数转化为一棵树。

步骤 1　令 P 为原始 prüfer 数，Q 为所有不包含在 P 的端点的集合。

步骤 2　将 Q 中的节点标号数字从左到右升序排列，保证 Q 的最左边的数字即为最小数字。令 i 为 Q 中的最左边的数字（即最小数字），j 为 P 的最左边的数字。将边 (i,j) 添加到树上，将 i 从 Q 中除去，j 从 P 中除去。若 j 在 P 的剩余部分中不再出现，将 j 加入到 Q 中。重复以上过程，直到 P 中无数字。

步骤 3　若 P 中已无数字，Q 中只有两个合格节点 i 和 j，将边 (i,j) 加入到树中，于是形成 $n-1$ 条边的树。

3.2　初始化种群和度的改进

prüfer 数（染色体）的初始化由 $[1,n]$ 中随机产生的 $n-2$ 个整数开始执行，其中 n 为树的节点个数。这样就可能产生不可行的染色体，因此还需对不满足度约束的染色体进行改进。除此之外，在交叉和变异的后代中都会出现不满足度约束的情况。

运用 prüfer 数编码能较方便地检验染色体是否可行，可以根据该编码法包含节点度的信息进行判断。检验条件定为：节点在 prüfer 数中出现的次数不超过其规定的度的上限 $d-1$。如果所有节点满足检验条件，即满足度约束条件；否则，需进行改进。

改进方法：如果一个节点在 prüfer 数编码中出现的次数超过 $d-1$，违反了度约束，那么将多出的点随机替换为未在编码中出现的节点。

3.3　基本遗传算子

遗传操作是模拟生物基因遗传的操作。在遗传算法中，通过编码组成初始群体后，遗传操作的任务就是对环境适应的程度（适应度评估）施加一定的操作，从而实现优胜劣汰的进化过程。从最优搜索的角度而言，遗传操作可使问题的解，逐代优化，并逼近最优解。

遗传操作包括三个基本遗传算子（genetic operator）：选择（selection）、交叉（crossover）、变异（mutation）。这三个遗传算子有如下特点：

（1）这三个遗传算子的操作都是随机扰动情况下进行的。

（2）遗传操作的效果和上述三个遗传算子所取的操作概率、编码方法、群体大小、初始群体以及适应度函数的设定密切相关。

（3）其操作方法或操作策略随着具体求解问题的不同而异，与个体的编码方式直接相关。

本文采用的 $(\mu+\lambda)$ 选择是确定性的选择方案，即从父代和子代中选取最好的染色体，从而确保最优染色体能够在新一代中得到保留。均匀交叉（Uniform Crossover）是两个配对个体的每一个基因以相同的交叉概率进行交换，是遗传算法区别于其他进化算法的主要特征，是产生新个体的主要方法[8]，均匀交叉可用图 1 表示。变异操作采用了随机扰动变异（Random Perturbation Mutation）。即随机选择一个位置并随机替代一个 $[1,n]$ 中的整数。

$$\begin{matrix} A: x\ x\ x\ x\ x\ x\ x\ x\ x\ x & \xrightarrow{W=0101010101} & A': x\ y\ x\ y\ x\ y\ x\ y\ x\ y \\ B: y\ y\ y\ y\ y\ y\ y\ y\ y\ y & & B': y\ x\ y\ x\ y\ x\ y\ x\ y\ x \end{matrix}$$

图 1　节点的均匀交叉

3.4　评价

遗传算法可以通过使用目标函数值得到下一步的有关搜索信息，对目标函数值的使用是通过评价个体的适应度来体现的。评价的一般过程：对个体编码进行解码处理（将染色体转换为树）；计算对应个体的目标函数值（计算树的总权）；由目标函数值按一定规则求出个体的适应度。按进化规则，适应度越大的个体，其进化繁殖后代的机会也就越大，有利于在编码空间寻找最适合的编码结构和解空间的最优解。

3.5　求解 DCMST 问题的遗传算法

求解 DCMST 问题的遗传算法是在简单遗传算法中加入了节点度的修改过程，使其满足度约束。其基本过程可表示如下：

```
begin
    迭代 t←0；初始化 P(t)；修改 P(t)的度；评价 P(t)；
    while(终止条件不满足)do
        执行交叉、变异操作 P(t)以产生子代 C(t)；
        修改 C(t)的度；评价 C(t)；
        从 P(t)和 C(t)选择 P(t+1)；t←t+1；
    end
end
```

4　DCMST 问题解的上下界

一般情况下，无度约束的最小树问题的解可作为 DCMST 问题的下界（lower bounds），本文采用 Prim 算法得到这个下界。

为了获得 DCMST 问题的上界（upper bounds），本文设计了一种边边交换的启发式算法（Heuristic Algorithm，HG），其基本思想是：先找到 MST，若不满足度约束，则交换边使其满足度约束。考虑到 Prim 算法可得到 MST，于是改进 Prim 算法以获得 DCMST 问题的上界。步骤如下：

步骤 1　利用 Prim 算法求 MST，并计算其权重。（可以作为 DCMST 问题的下界）

步骤 2 检查 MST 是否满足度约束，如果 MST 满足度约束，则算法停止；否则，利用边边交换的方法改进 MST 中不满足度约束的节点。

检查 MST 是否满足度约束的具体方法：Edge 中所有节点标号出现的次数即位节点的度，只要判断节点出现次数是否超过度约束即可。

边边交换的具体方法：找到 MST 中所有不满足度约束的节点 v_i 和度值小于度约束的节点 v_j，分别加入到 UD 和 D 中，将 Edge 中多出的点 v_i 随机替换为 D 中的点 v_j，并将该点从 D 中删去，直到 Edge 中所有的点都满足度约束。这样处理很容易使度值满足约束条件，但可能出现重复边的现象，所以在程序设计中需多次执行边边交换，以排除此现象。

5　数值分析

为验证 GA 求解 DCMST 问题的可行性和有效性，算法采用 C 语言和 MATLAB 的混合编程实现。

例 1　给定图 G 的权矩阵 $\boldsymbol{W}$（取自文献[5]），各节点的度约束为 $b_i=3(i=1,2,\cdots,9)$。

$$\boldsymbol{W}=\begin{bmatrix} +\infty & 3 & 3 & 5 & 16 & 5 & 12 & 21 & 23 \\ 3 & +\infty & 2 & 2 & 9 & 4 & 7 & 18 & 19 \\ 3 & 2 & +\infty & 7 & 13 & 9 & 15 & 22 & 24 \\ 5 & 2 & 7 & +\infty & 7 & 2 & 2 & 12 & 14 \\ 16 & 9 & 13 & 7 & +\infty & 15 & 9 & 20 & 11 \\ 5 & 4 & 9 & 2 & 15 & +\infty & 4 & 10 & 17 \\ 12 & 7 & 15 & 2 & 9 & 4 & +\infty & 6 & 8 \\ 21 & 18 & 22 & 12 & 20 & 10 & 6 & +\infty & 10 \\ 23 & 19 & 24 & 14 & 11 & 17 & 8 & 10 & +\infty \end{bmatrix}$$

一般情况下，图 G 的满足度约束的最小生成树并不唯一，应用遗传算法得到该例的六种 DCMST，分别对应各自唯一的 prüfer 数编码，如图 2 所示。

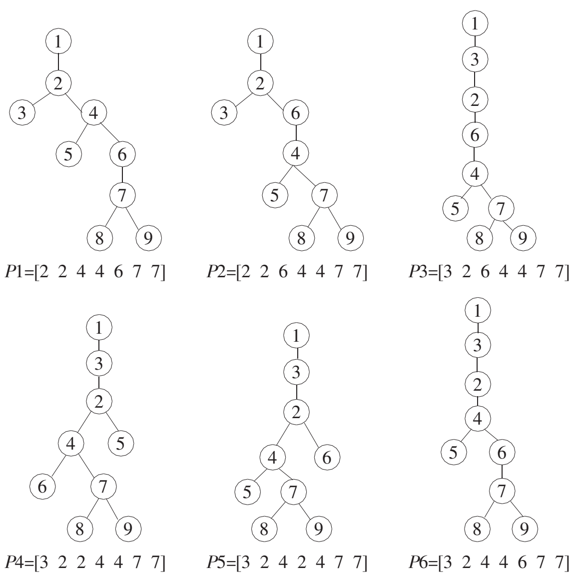

图 2　DCMST 的 prüfer 数编码及与之相对应的树

应用遗传算法得到 DCMST 的最优解为 34。遗传算法的参数设置为：种群大小为 50；交叉操作的概率为 0.2；变异操作的概率为 0.05；迭代世代数为 200。该例应用快速近似算法得到 DCMST 的近似最优解为 36[5]，应用分支定界法得到的最优解为 34[9]。

如果各节点的度约束为 $b_i=2(i=1,2,\cdots,9)$，该问题即为货郎担问题（TSP）。该例应用快速近似算法得到 DCMST 的近似最优解为 40[5]，应用分支定界法[9]和本文的遗传算法得到DCMST 的最优解为 39。

例 1 是一个规模较小的 DCMST 问题，应用遗传算法和分支定界法都能够快速准确地解决这个问题，快速近似算法的结果也比较令人满意。但当问题规模变大时，快速近似算法显然不能得到比较准确的结果，而且其结果也将越来越偏离最优值。分支定界法虽能够找出最优解，但算法本身的特点决定其花费很多内存空间，当内存容量有限、问题规模扩大时，此方法带来的问题会日益显著。对于中等及以上规模的问题，精确的算法和许多启发式算法的效率很低，而进化算法却非常有效[8]。鉴于此，将运用遗传算法求解以下规模较大的例子并说明算法的有效性。

例 2　为了验证遗传算法的有效性，首先，绘制了节点数为 10，节点间的距离或费用值随机产生并均匀分布于区间[10，100]上，度约束取 3 的数值例子的解分布图（图 3）。在程序运行 50 次中，得到最优解的次数所占百分比为 58%。

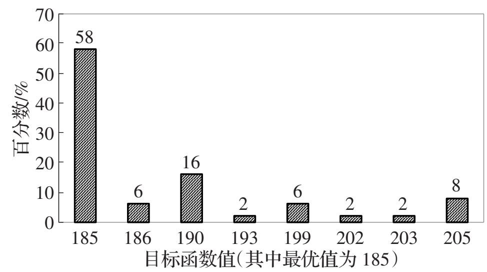

图 3　遗传算法求解 DCMST 问题的解分布图

构造了 8 个数值例子，节点数分别从 15 个到 30 个，节点间的距离或费用值随机产生并均匀分布于区间[15，30]上，度约束分别取 3、4。遗传算法的参数设置为：种群大小为 500；交叉操作的概率为 0.3；变异操作的概率为 0.05；迭代世代数为 1 000。为了防止遗传算法的随机性对结果的影响，程序运行 20 次，计算结果及其分析如表 1、表 2 所示。

表 1　GA 的结果分析与 CPU 计算时间

问题规模（节点数）	DC	$Best(C)$	$Worst(C)$	$Avg.(C)$	$\sigma(C)$	CPU/s
15	3	219	226	221.20	1.755 4	58
	4	219	222	219.90	1.165 3	57
20	3	293	304	298.30	3.213 5	82
	4	292	303	297.50	2.781 5	80
25	3	371	392	383.10	5.729 9	108
	4	370	387	379.70	4.378 2	108
30	3	454	472	465.50	4.915 1	142
	4	452	474	465.00	5.785 6	141

表 2　度约束为 3 的结果与上下界的比较

问题规模（节点数）	LB（without DC）（Prim）	UB（HG）	$Best$（PGA）	$(Best-LB)/LB$ /%	$(UB-Best)/UB$ /%
15	219（3）	221	219	0.00	0.90
20	291（4）	294	293	0.69	0.34
25	365（4）	374	371	1.64	0.80
30	444（4）	454	454	2.25	0.00

表 1 中，DC 表示度约束；$Best(C)$ 表示程序运行 20 次中目标函数的最优值；$Worst(C)$ 表示最差的值；$Avg.(C)$ 表示平均最优值；σ 是标准差，表示计算结果的离散程度。从均值 $Avg.$ 可以

看出，遗传算法求解 DCMST 问题能够得到比较理想的最优解近似最优解；而且标准差 σ 在 6%以内，说明本文的算法具有良好的稳定性；CPU 的计算时间在 2.5 min 以内，算法具有比较高的效率。表 1 的结果还说明度约束较紧时，同样规模问题的计算量就要增加，CPU 的计算时间也就随着增加，反之亦然。表 2 中，*LB* 代表下界，由 Prim 算法求得的 MST 问题的解，括号中是其最大度值；*UB* 代表上界，由边边交换的启发式算法（HG）求得的 DCMST 问题的解；*Best* 对应于表 1 中的最优解，由遗传算法求得。与下界的比较结果表明，遗传算法有效地改进了利用 HG 所求得的结果，从另一方面验证了算法的有效性。

以下是节点数分别为 40 到 120 的 5 个数值例子，节点间的距离或费用值随机产生并均匀分布于区间[10，100]上，度约束取 3。遗传算法的参数设置为：种群大小为 500；交叉操作的概率为 0.4；变异操作的概率为 0.05；迭代世代数为 3 000。程序运行 10 次，计算结果如表 3 所示。随着规模问题的扩大，CPU 的计算时间将有较大幅度的增加，但对于大规模的 DCMST 问题而言，这样的计算时间是可以接受的，而且结果与上下界的比较也令人满意。因此，本文所用的遗传算法仍是求解 DCMST 问题的有效算法。

表 3 GA 的结果分析与 CPU 计算时间

问题规模（节点数）	*LB*（without *DC*）（Prim）	*UB*（HG）	*Best*（PGA）	(*Best*–*LB*)/*LB* /%	(*Best*–*UB*)/*UB* /%	CPU /s
40	504（5）	631	586	16.27	7.13	631
60	662（5）	814	798	20.54	0.20	1 334
80	877（5）	1 350	1 222	39.34	9.48	2 198
100	1 053（5）	1 496	1 471	39.70	1.67	3 136
120	1 254（5）	1 712	1 610	28.38	5.96	4 515

6 结束语

DCMST 问题在信息网络的设计与优化中都有十分重要的应用背景，由于该问题属于 NP-完全问题，至今未有十分有效的最优算法。传统的启发式算法、分枝定界法等只能求解较小规模的带约束最小树问题，当目标函数和约束条件种类繁多，并随着问题规模的扩大，要寻求到一种能以有限的代价来解决这些最优化问题的通用方法仍是一个难题。遗传算法为人们解决这类问题提供了一个有效的途径和通用框架，开创了一种新的全局优化搜索算法。

参考文献：

[1] 玄光男，程润伟.遗传算法与工程优化[M].北京：清华大学出版社，2004：67，231-233.

[2] Narula S，Ho C.Degree-constrained minimum spanning tree[J].Computers and Operations Research，1980，7：239-249.

[3] Volgenant A.A Lagrangean approach to the degree-constrained minimum spanning tree problem[J].European Journal of Operational Research，1989，39：325-331.

[4] 马良，蒋馥.度约束最小生成树的快速算法[J].运筹与管理，1998，7（1）：1-5.

[5] 宋海洲.求解度约束最小生成树的快速近似算法[J].系统工程学报，2006，21（3）：232-236.

[6] Zhou G，Gen M.Approach to the degree-constrained minimum spanning tree problem using genetic algorithms[J].Engineering Design and Automation，1997，3（2）：157-165.

[7] Zhou G，Gen M.A note on genetic algorithm approach to the degree-constrained spanning tree problems[J].Networks，1997，30：91-95.

[8] Raidl G R，Julstrom B A.Edge-sets：an effective evolutionary coding of spanning trees[J].IEEE Transactions on Evolutionary Computation，2003，7（3）：225-239.

[9] 顾立尧.带有度约束的最小耗费生成树的分支限界算法[J].计算机应用与软件，1989，6（6）：49-54.

（上接 52 页）

$$\lambda=-\frac{|\boldsymbol{R}|}{\sum\limits_{k=1}^{M}|(\boldsymbol{R}_1,\boldsymbol{R}_2,\cdots,\boldsymbol{R}_{k-1},\boldsymbol{1},\boldsymbol{R}_{k+1},\cdots,\boldsymbol{R}_M)|} \tag{15}$$

将式（15）代入（14）得到方程组的解为：

$$\alpha_i^*=\frac{|(\boldsymbol{R}_1,\boldsymbol{R}_2,\cdots,\boldsymbol{R}_{i-1},\boldsymbol{1},\boldsymbol{R}_{i+1},\cdots,\boldsymbol{R}_M)|}{\sum\limits_{k=1}^{M}|(\boldsymbol{R}_1,\boldsymbol{R}_2,\cdots,\boldsymbol{R}_{k-1},\boldsymbol{1},\boldsymbol{R}_{k+1},\cdots,\boldsymbol{R}_M)|}$$

$$\alpha_i^*=\frac{(R_{1i}+R_{2i}+\cdots+R_{Mi})}{\sum\limits_{k=1}^{M}|(\boldsymbol{R}_1,\boldsymbol{R}_2,\cdots,\boldsymbol{R}_{k-1},\boldsymbol{1},\boldsymbol{R}_{k+1},\cdots,\boldsymbol{R}_M)|}\quad i=1,2,\cdots,M \tag{16}$$

其中 R_{qi} 是矩阵 $\boldsymbol{R}$ 中 r_{qi} 的代数余子式。

为了进一步理解 $\alpha_i^*(i=1,2,\cdots,M)$，考虑 $\boldsymbol{R}$ 的逆矩阵 $\boldsymbol{R}^{-1}$。由逆矩阵定义可以得到，$\boldsymbol{R}^{-1}=\frac{\boldsymbol{R}^*}{|\boldsymbol{R}|}$，其中 $\boldsymbol{R}^*$ 是 $\boldsymbol{R}$ 的伴随矩阵。则 $\boldsymbol{R}^{-1}$ 中元素为 $\frac{R_{ij}}{|\boldsymbol{R}|}$，其中 R_{ij} 是矩阵 $\boldsymbol{R}$ 中 r_{ij} 的代数余子式。因此，α_i 的最优值 α_i^* 为：

$$\alpha_i^*=\frac{\sum\limits_{j}R_{ij}^{-1}}{\sum\limits_{k}\sum\limits_{j}R_{kj}^{-1}} \tag{17}$$

将式（17）代入（10）后，得到估计器的线性融合最优泛化误差为：

$$GErr(f_{gens}^{(M)})=<\sum_{m=1}^{M}\sum_{l=1}^{M}\alpha_m^*\alpha_l^*R_{ml}>_{x_0} \tag{18}$$

当 $\alpha_1=\alpha_2=\cdots=\alpha_M=\frac{1}{M}$ 时，泛化误差式（18）则是等概率意义上的泛化误差分解。

4 结语

在机器学习中，为了研究估计器的特性，通常采用泛化误差分解的方法。针对加权融合方法与平方误差损失函数，给出了泛化误差分解的推导过程，在此基础上，进一步获得了加权融合方法的最优泛化误差分解。然而，在实际问题中，由于问题的性质不同，使用平方误差损失函数不一定是好的选择方法，于是研究一般意义上的泛化误差分解是需要进一步研究的内容。

参考文献：

[1] Geman S，Bienenstock E，Doursat R.Neural networks and the bias/variance dilemma[J].Neural Computation，1992，4（1）：1-58.

[2] Domingos P.A unified bias-variance decomposition for zero-one and squared loss[C]//Proc of ICAI，2000.

[3] Ueda N，Nakano R.Generalization error of ensemble estimators[C]//Proc of ICNN，1996.