求解01背包问题的贪婪蛙跳算法

高思齐 邢玉轩 肖 侬 刘 芳
(国防科技大学计算机学院 长沙 410073)

摘 要 01背包问题是经典的组合优化问题,被广泛应用于生活中的多个领域,如货物装载、预算控制、资源分配和资产管理等。因此,长期以来许多科学家在该领域不断钻研,并取得了丰硕的成果。尽管01背包问题已被研究多年,但由于该问题已被证明为NP完全问题,因此找到最优解并不容易。近年来,大量的智能算法不断被提出并被用来求解01背包问题,如化学反应优化算法、遗传算法、粒子群算法、蛙跳算法、人工蜂群算法、爬山算法和模拟退火算法等。通过对智能算法和01背包问题的探索,文中提出了贪婪蛙跳算法(GFLA)来解决01背包问题。不同于传统的蛙跳算法,GFLA总会在每次模因搜索过程中更新全局最优解,以便在接下来的全局搜索过程使用最新的全局最优解进行搜索,从而扩大解的搜索空间。除了蛙跳算法这类传统的局部搜索和全局搜索策略之外,针对01背包问题,在计算适应度值的阶段,本工作提出了贪心策略并分别将其应用于drop和add两个步骤。在drop阶段,若背包超重,则将其中价值密度最小的物品移出并更新解决方案。在add阶段,若背包还有承载物品的能力,则将未放入背包的重量最小的物品放入背包,并对背包信息进行更新。这样,便大大提高了利用蛙跳算法来求解01背包问题的能力。将贪婪蛙跳算法与蜂群算法、化学反应优化算法、遗传算法和量子演化算法进行对比,结果显示,贪婪蛙跳算法取得了最好的结果,从而表明了该算法是求解01背包问题的有效算法。

关键词 01背包问题,贪婪策略,价值密度,最小分配,蛙跳算法

中图法分类号 TP301 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.07.011

Greedy Frog Leaping Algorithm for 01 Knapsack Problem

GAO Si-qi XING Yu-xuan XIAO Nong LIU Fang
(School of Computer, National University of Defense Technology, Changsha 410073, China)

Abstract 01 knapsack problem(01kp) is a classic combinatorial optimization problem and appears in many models of real world problem, such as cargo loading, budget control, resource allocation, financial management, and so forth. Many scientists have studied on this area for several decades and achieved a rich harvest. Even though 01kp has been researched for many years, it has been proved to be an NP complete problem, thus finding the best solution is not easy. In recent years, many original and improved intelligent algorithms was proposed to address 01 knapsack problem, for instance, chemical reaction optimization algorithm, genetic algorithm, particle swarm optimization algorithm, shuffled frog leaping algorithm, artificial bee colony algorithm, hill climbing algorithm and simulated annealing algorithm. This paper studied on many intelligent algorithms and 01 knapsack problems, and proposed greedy frog leaping algorithm (GFLA) to solve 01kp. Different from original shuffled frog leaping algorithm, GFLA always updates global best solution during each memeplex process, such that global exploration would employ the latest global best solution to expand the search space. In addition to traditional global exploration and local exploitation, aiming at 01kp, it proposed a greedy scheme at fitness computing stage, including drop and add two steps. At drop step, if the knapsack is over-weighted, then the item with minimum value density is removed from the knapsack and the solution is updated. At add step, if there is still space for the knapsack to load items, then items with minimum weight which have not been loaded are put into the knapsack and solution is also updated. Drop and add steps adopted in our work greatly improve greedy frog leaping algorithm's ability to address 01kp. This paper conducted the experiments on benchmarks, and compared the results with chemical reaction optimization algorithm, genetic algorithm, quantum-inspired evolutionary algorithm, chemical reaction optimization with greedy strategy. All experimental results show that greedy frog leaping algorithm obtains the best solutions in all cases, which indicates that GFLA is an efficient algorithm to address 01 knapsack problem.

Keywords 01 knapsack problem, Greedy scheme, Value density, Min-allocate, Frog leaping algorithm

到稿日期:2017-07-18 返修日期:2017-10-24 本文受国家高技术研究发展计划(863),国家自然科学基金项目(2015AA015305,61232003,61332003,61202121)资助。

高思齐(1994-),女,硕士,主要研究方向为群体智能算法和神经网络,E-mail:gsq_fighting@126.com;**邢玉轩**(1990-),男,博士,主要研究方向为大数据处理;**肖 侬**(1969-),男,教授,主要研究方向为云计算、存储以及计算机体系结构,E-mail:nongxiao@nudt.edu.cn;**刘 芳**(1976-),女,副教授,主要研究方向为网络存储和信息安全,E-mail:liufang@nudt.edu.cn(通信作者)。

1 引言

人类生活的很多方面都受决策结果的影响，如预算控制、资源分配和证券投资组合优化等。一个决策过程应该是量化并被独立可验证的，如此才能衡量某一决策的优劣程度。因此，对影响决策的所有因素进行量化并对可行解进行分析十分重要。一般情况下，决策输出可由单个数值来表示，如收益、损失和开销等。问题求解方案的不同组合方式影响着可行解的结果及决策输出。为了对决策问题进行更深层次的探索和认知，文中选用 01 背包问题进行研究。

01 背包问题是组合优化问题中经典的 NP 问题之一，其 NP 完全性的证明可参考文献[1]。01 背包问题是货物装载及任务调度等问题的抽象，因此对该问题的研究十分重要。通过研究发现，元启发式算法在求解 01 背包问题的场景中有着较好的效果，因此我们选用蛙跳算法来求解 01 背包问题并对算法进行优化。

给出 01 背包问题的描述如下：给定 n 个物品，每个物品都有自己的重量 w_i 和价值 p_i，现有一个承重能力为 C 的背包，要求选中多个物品并将其放入背包中。问题是怎样选择物品的组合，以在保证背包承重能力的情况下，尽可能使背包中物品的价值总和最大。其数学模型为：

$$\begin{aligned}&\text{maximize } \sum_{i=1}^{n} x_i p_i \\ &\text{subject to } \sum_{i=1}^{n} x_i w_i \leqslant C, x_i \in \{0,1\}, \forall i \in \{1,2,\cdots,n\}\end{aligned} \tag{1}$$

由于物品不可分割，因此采用 01 变量 x_i 来表示物品是否被放入背包。$x_i=1$，表示物品被放入背包；$x_i=0$，表示物品未被放入背包。

本文提出了贪婪蛙跳算法来解决 01 背包问题。不同于传统蛙跳算法的全局搜索过程，优化后算法的全局最优解在每次的模因组演化之后都会及时得到更新。此外，文中还提出了贪婪策略，用于背包问题的修正操作。修正操作包含两个阶段：add 阶段和 drop 阶段。若背包中物品重量的总和超过背包重量的限制，则在 drop 阶段不断将背包中价值密度最小的物品移出背包；若背包仍有承重空间，则在 add 阶段将背包外重量最小的物品不断放入背包。这种方法被称为最小分配策略。采用最小分配策略可有效地扩大算法的搜索空间，并取得很好的求解效果。本文第 2 节介绍 01 背包问题的相关研究工作；第 3 节阐述传统蛙跳算法的基本思想；第 4 节深入分析本文提出的贪婪蛙跳算法；第 5 节给出实验结果及性能分析；最后总结全文并探讨贪婪蛙跳算法的应用场景及未来的优化方向。

2 相关工作

整体来看，解决 01 背包问题的方案大致可分为两种：确切算法(Exact Technology)和元启发式算法。确切算法包括动态规划、分支界定算法及核心算法(Core Algorithm)。元启发式算法包括化学反应优化算法、蚁群算法、人工蜂群算法、遗传算法、蛙跳算法和粒子群算法等。将大多数确切算法用于解决问题的核心(Core)，然后将部分解决方案应用到全局解中。然而，针对特定问题的确切算法很难被扩展并使用到其他问题中，内存消耗也较严重，因此许多元启发式算法不断被研究并改进。在早期的元启发式算法中，针对背包问题的“多项式近似策略”由 Sahni 首次提出[2]。在这种方案中，解的质量可在求解之前有一个期望值。随着求解质量期望值的提高，不同的求解方法也逐渐涌现。多年来，许多用于解决 01 背包问题的智能算法被提出。Shi 等[3]使用蚁群算法解决 01 背包问题；Yan 等[4]将化学反应优化算法和禁忌搜索相结合，以扩大解的搜索空间；Han 等[5]提出了量子演化算法；Shen 等[6]提出了用于解决 01 背包问题的遗传算法；Truong 等[7]提出了基于贪心策略的化学反应优化算法。许多用于求解 01 背包的智能算法均提供了可能的解决方案，但一些算法由于自身的设计可能会陷入局部最优解，如原始的蚁群算法。原始的蚁群算法中，解的效果越好，其信息素值就会越高，其他个体也会沿着信息素的浓度寻求新的解决方案，从而很容易使种群陷入局部最优解。因此，随后又出现了 min-max ACO 等改进算法的设计。

考虑到以上因素，本文提出用贪婪蛙跳算法来求解 01 背包问题。原始蛙跳算法和改进之后的算法分别在第 3 节和第 4 节进行阐述。

3 原始混合蛙跳算法

原始的混合蛙跳算法(Shuffled Frog Leaping Algorithm，SFLA)由 Eusuff 等[8]提出，用于解决组合优化问题的模因启发式算法。作为一种基于种群的协作搜索模型，SFLA 是受自然界中模因组的启发而提出的，包括局部搜索过程和全局的信息交换。SFLA 由相互作用的青蛙个体组成，将每只青蛙划分到不同的模因组(memeplex)，并针对每个模因组同时进行独立的局部搜索。为进行全局搜索，定期对蛙群进行洗牌并将其重组为新的模因组。除此之外，为跳出局部最优解，SFLA 会在种群演化的过程中随机生成新个体。

为形象化地解释 SFLA，考虑一组青蛙在沼泽中跳跃的情景。沼泽中有许多石头分布在不同的位置，每个石头上都有数量不等的食物；青蛙在沼泽中跳跃，并尽可能地寻找拥有较多食物的石头；青蛙个体之间可以互相交流，以利用其他个体的信息来更新和改进自己的模因。模因的改进是通过改变每只青蛙的跳跃步数来修正个体青蛙的位置。通过种群的不断迭代演化和信息交流，蛙群最终会找到食物最多的石头。

SFLA 是一种确定性与随机性相结合的方法。确定性策略允许算法有效地使用响应信息来指导启发式搜索，而随机因素则保证搜索模式的鲁棒性和灵活性。搜索从随机生成的青蛙种群开始。种群被划分为若干个平行的模因组，每个模因组可在不同的空间独立演化。在每个模因组内部，每只青蛙会受到其他青蛙个体的影响，因此会经历模因进化。模因进化会提高每个个体的模因质量，进而使个体朝着目标演进。为保证优胜劣汰，拥有较好模因的青蛙个体对种群的演化影响更大。在进化过程中，每个模因组会保存青蛙子群的局部最优值，并使用它来对该子群中的最差个体进行更新。每次迭代完成之后，将所有模因组混合以形成新的种群，并从中选出整个种群的最优解以对种群的最差个体进行更新。种群的重新洗牌使得来自不同模因组的青蛙个体相互影响，使不同个体之间的信息共享，有利于提高解的质量。

4 贪婪蛙跳算法

4.1 算法流程

针对 01 背包问题的贪婪蛙跳算法大体包含 3 个部分：局

部搜索、全局搜索和修正操作。

算法的具体步骤如下：

Step1　初始化。设种群大小为 P，模因组的数量为 m，每个模因组中青蛙的个数为 f，且满足 $P=m*f$。

Step2　随机生成初始种群。在可行解空间内生成 P 只青蛙个体，第 i 只青蛙个体可用向量 $V(i)$ 表示：$V(i)=(V_{i1},V_{i2},\cdots,V_{in})$。其中，$n$ 为背包问题中的物品数量；$V_{ij}=0$ 或 1，$j=1,2,\cdots,n$。

对于每一只青蛙，按照式(1) 计算其适应度值 $fitness(i)$。

Step3　适应度值排序。将 P 只青蛙按照适应度值的大小降序排列，得到适应度值下标数组：$FitnessSub=\{fitsub(0),fitsub(1),\cdots,fitsub(f-1)\}$。其中，$fitsub(0)$所标识的青蛙的适应度值最大，即 $fitness(fitsub(0))\geqslant fitness(fitsub(i))$，$\forall i\in\{0,1,\cdots,f-1\}$。

例如，假设种群中有 5 只青蛙，其适应度值的排序为：$fitness(2)>fitness(1)>fitness(0)>fitness(4)>fitness(3)$。其中，$fitness(i)$中的 i 代表第 i 只青蛙。则 $fitsub(0)=2$，$fitsub(1)=1$，$fitsub(2)=0$，$fitsub(3)=4$，$fitsub(4)=3$。

Step4　模因分组。将 f 只青蛙按照适应度值的大小依次放到 m 个模因组中。

如 Step3 中的例子，假设有 3 个模因组，$fitsub(0)$放入第 0 个模因组，$fitsub(1)$放入第 1 个模因组，$fitsub(2)$放入第 2 个模因组，$fitsub(3)$放入第 0 个模因组，$fitsub(4)$放入第 1 个模因组，依此类推。

Step5　模因演化。对每一个模因组进行模因演化操作。设 m 为模因组的个数，$memIters$ 为每个模因组的迭代次数。将每个模因组中性能最好和最差的个体分别表示为 P_{lb} 和 P_{lw}，整个种群中性能最好的个体表示为 P_{gb}。在每一轮的迭代过程中，均对模因组中性能最差的个体进行更新。

01 背包中的模因演化算法见算法 1。每次执行模因演化算法时都会先获取种群的全局最优解。

Step6　模因组混合。在每个模因组内进行一定数量的模因演化之后，将 m 个模因组重新洗牌以形成新的种群。依然将种群个体按照适应度值降序排列，并更新种群中的最优个体 P_{gb}。

Step7　算法结束条件的判定。若满足算法终止条件，如算法迭代了指定的次数或算法已收敛，则退出程序；否则转至 Step4 继续执行。

算法 1　模因演化算法

```
obtain global best frog as Pgb
  for i←0 to m−1 do
    obtain the i-th memplex as memple xi
    for iter←0 to memIters−1 do
      obtain the local best frog in memplexi as Plb
      obtain the local worst frog in memplexi as Plw
      distance←rand() * (Plb−Plw)
      Ptemp←Plw+distance
      if(fitness(Ptemp)>fitness(Plw)) then Plw←Ptemp
      else
        distance←rand() * (Pgb−Plw)
        Ptemp←Plw+distance
        if(fitness(Ptemp)>fitness(Plw)) then Plw←Ptemp
        else
          randomlyInit(Plw)
        endif
      endif
    endfor
    update Pgb
endfor
```

4.2　适应度值的计算

种群中青蛙的适应度值的计算公式为：

$$fitness=\sum_{i=1}^{n}x_ip_i \tag{2}$$

其中，n 为物品个数；p_i 为第 i 个物品的价值；$x_i=0$ 或 1，表示第 i 个物品是否放入背包。

除了进行适应度值的计算以外，还要对求解方案进行修正操作，因为物品的总重量和可能会超过背包的承载能力 C。因此，本文采用不同的贪婪策略分别在 drop 阶段和 add 阶段对解决方案进行修正。

(1)drop 阶段

首先介绍价值密度的概念。价值密度，即 value density，用于衡量物品单位重量的价值。

$$value\ density_i=p_i/w_i$$

其中，p_i 为第 i 个物品的价值，w_i 为第 i 个物品的重量。

在计算适应度值之后，首先判断背包中物品重量之和 $sumWeight$ 是否超出了背包的承重能力 C。若 $sumWeight>C$，则不断地把背包中价值密度最小的物品移出，直至满足背包承载能力的限制为止。

(2)add 阶段

drop 阶段完成之后，为得到最好的解，不断地把背包外的物品放入背包，直到达到背包承重能力的极限为止。本文先将物品按照 $weight$ 的大小进行升序排列，然后依次遍历物品。若此时背包中物品的总重量为 $sumWeight$，物品 i 未被放入背包，且 $w_i+sumWeight\leqslant C$，则将 x_i 置为 1，同时对解的 $fitness$ 值进行更新。

贪婪蛙跳算法的流程图如图 1 所示。

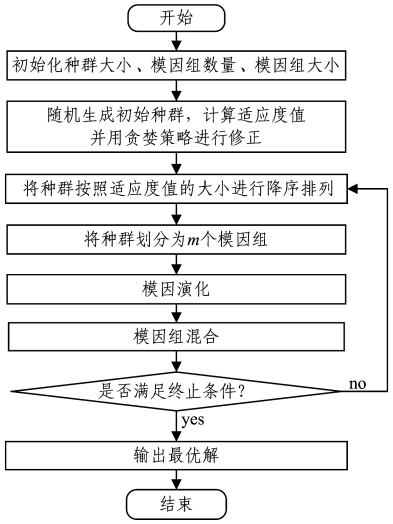


图 1　GFLA 算法流程图

Fig. 1　Flowchart of GFLA algorithm

5 实验

为测试贪婪蛙跳算法的性能，在随机生成的 01 背包数据集上进行实验，并将该算法与蚁群算法、遗传算法、人工蜂群算法、量子演化算法和贪婪化学反应优化算法进行比较。

在所有的测试用例中，01 背包数据集的生成方式如下：

$w_i = \text{rand}[1,10]$

$p_i = w_i + 5, i = 1,2,\cdots,n$

$C = \frac{1}{2}\sum_{i=1}^{n} w_i$

其中，rand[1,10]表示在整数[1,10]之间使用均匀分布随机地生成物品重量。

人工蜂群算法中，种群大小为 100，其中 employed bee 和 onlooker bee 的个数各为 50。贪婪蛙跳算法的种群大小为 100，模因组的个数为 10，每个模因组进行模因演化时的迭代次数为 10。

人工蜂群算法和贪婪蛙跳算法均由 C++实现，实验环境为 centos 操作系统，设备上包含 2 块 Intel CPU，规格为 Intel (R) Xeon(R) CPU E5-2620 0 @ 2.00GHz，另有 16GB RAM。

蚁群算法、遗传算法、量子演化算法和贪婪化学反应优化算法的实验结果来自文献[7]。所有算法的结果均取运行 30 次的平均值，每次运行中种群进行了 1000 次迭代。

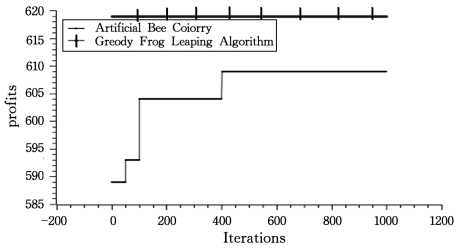
不同算法在 3 个随机生成的数据集上的实验结果如表 1 所列。

表 1 不同数据集上的实验结果

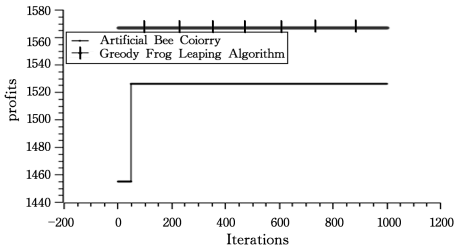
Table 1 Results on different datasets

	100 items			250 items			500 items		
	best solution	average	time/(s/run)	best solution	average	time/(s/run)	best solution	average	time/(s/run)
ACO	585	581.2	1.015	1530	1523.6	2.301	2930	2917.4	4.650
GA	585	582.9	1.102	1535	1524.9	2.475	2935	2921.3	4.968
QEA	587	583.2	0.906	1537	1526.8	2.502	2935	2925.5	4.572
CROG	590	584.1	0.814	1539	1529.7	2.276	2940	2928.9	4.386
ABC	609	605.5	0.173	1526	1509.3	0.393	2970	2935.3	0.736
GFLA	**619**	**619.0**	0.511	**1567**	**1567.0**	1.176	**3116**	**3115.9**	2.204

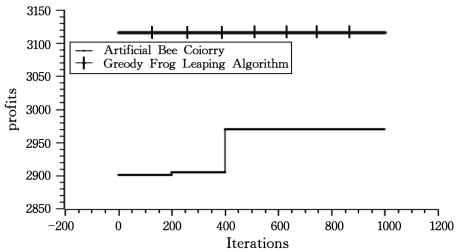
为比较算法的收敛性能，对人工蜂群算法和贪婪蛙跳算法的迭代求解情况分别进行了比较，结果如图 2 所示。



(a)100 个数据项实例上的收敛曲线



(b)250 个数据项实例上的收敛曲线



(c)500 个数据项实例上的收敛曲线

图 2 人工蜂群算法和贪婪蛙跳算法的收敛性比较

Fig. 2 Convergence comparison between ABC and GFLA

由以上实验结果可以看出，贪婪蛙跳算法在所有的测试集上均取得了最好的结果；且除了人工蜂群算法的运行时间最短之外，贪婪蛙跳算法每次运行的执行时间也是较优的。通过与人工蜂群算法的收敛性结果对比可以看出，贪婪蛙跳算法总是能在最短的时间内找到最优的解决方案。因此，贪婪蛙跳算法是求解 01 背包问题的有效算法。

结束语 本文提出了贪婪蛙跳算法来解决 01 背包问题。除了传统的蛙跳算法的局部搜索和全局搜索之外，本文在解决方案的修正阶段采用了贪婪策略来扩大解的搜索空间。解的修正阶段包括两个步骤：drop 阶段和 add 阶段。当背包中的物品总重量超过了背包的承重能力限制时，不断地将背包中密度价值较小的物品移出背包；当背包仍有多余的承重能力时，将未放入背包的物品按照重量大小升序排列，然后依次将重量最小的物品放入背包，直至达到背包的承重能力极限为止。通过在不同数据集上的实验及与其他算法的对比结果可知，贪婪蛙跳算法总能取得最好的结果，并展现出优秀的收敛性。因此，贪婪蛙跳算法是解决 01 背包问题的有效算法。未来将会使用 GPU 加速贪婪蛙跳算法，并将其应用于云环境中的任务调度等场景。

参 考 文 献

[1] LAGOUDAKIS, MICHAIL G. The 0-1 knapsack problem--An introductory survey [Z/OL]. https://pdfs.semanticscholar.org/6bd6/2e0ba7233c5086fe4b9061926d191894714b.pdf.

[2] SAHNI S. Approximate Algorithms for the 0/1 Knapsack Problem[J]. Journal of the Acm, 1975, 22(1): 115-124.

[3] SHI H. Solution to 0/1 knapsack problem based on improved ant colony algorithm[C]// 2006 IEEE International Conference

on Information Acquisition. IEEE,2006:1062-1066.

[4] YAN C,GAO S,LUO H,et al. A hybrid algorithm based on tabu search and chemical reaction optimization for 0-1 Knapsack problem[C]// International Conference in Swarm Intelligence. Springer,Cham,2015:229-237.

[5] HAN K H,KIM J H. Quantum-inspired evolutionary algorithm for a class of combinatorial optimization[J]. IEEE Transactions on Evolutionary Computation,2002,6(6):580-593.

[6] SHEN W,XU B,HUANG J P. An Improved Genetic Algorithm for 0-1 Knapsack Problems[C]// International Conference on NETWORKING & Distributed Computing. IEEE Computer Society,2011:32-35.

[7] TRUONG T K,LI K,XU Y. Chemical reaction optimization with greedy strategy for the 0-1 knapsack problem[J]. Applied Soft Computing Journal,2013,13(4):1774-1780.

[8] EUSUFF M,LANSEY K,PASHA F. Shuffled frog-leaping algorithm:a memetic meta-heuristic for discrete optimization[J]. Engineering Optimization,2006,38(2):129-154.

(上接第 41 页)

效融合的效果,进而提升网络的表征效果。

综合上述实验结果:在基于矩阵分解的网络表征中,使用多种网络结构比使用单一网络结构学习到的网络表征能取得更好的效果,同时也证明了本文提出的融合了多种结构和文本信息的网络表征方法 MsTADW 有一定的优越性,尤其在有标注数据较少的情况下,该方法有更大的优势。

结束语　本文从两个方面对 TADW 方法进行了改进和深入分析。首先,通过大量实验结果证明了文本特征在分解式中的位置会影响网络表征的效果,如果网络中顶点的主题词数比较少,那么根据归纳矩阵分解的原理,文本特征 T 放在分解式中 Y 特征的位置时学习到的网络表征更高效。另外,针对社交网络的特殊性,提出了一种多结构及文本融合的网络表征方法 MsTADW,经与 DeepWalk 方法及 TADW 方法在分类任务性能上的对比可知,MsTADW 方法学习到的网络表征在进行多分类时准确率有了一定的提高;同时,MsTADW 方法在训练数据较少的情况下能取得更好的效果。进一步,我们可将本文方法扩展应用到其他具有多种结构的网络中,也将对其他应用场景中不同任务目标情况下的网络表征的学习方法进行更深入的研究。

参 考 文 献

[1] COYAL A,BONCHI F,LAKSHMANAN L. Learning influence probabilities in social networks[C]// ACM International Conference on Web Search & Data Mining. ACM,2010:241-250.

[2] CRANMER S,DESMARAIS B. Inferential Network Analysis with Exponential Random Graph Models[J]. Political Analysis,2011,19(1):66-86.

[3] FAN W. Graph pattern matching revised for social network analysis[C]// International Conference on Database Theory. ACM,2012:8-21.

[4] MIKOLOV T,SUTSKEVER I,CHEN K,et al. Distributed Representations of Words and Phrases and their Compositionality[J]. Advances in Neural Information Processing Systems,2013,26:3111-3119.

[5] KUANG L W,HAO F,YANG L,et al. A Tensor-Based Approach for Big Data Representation and Dimensionality Reduction[J]. IEEE Transactions on Emerging Topics in Computing,2017,2(3):280-291.

[6] CUNNINGHAM J,YU B. Dimensionality reduction for large-scale neural recordings[J]. Nature Neuroscience,2014,17(11):1500-1509.

[7] MEI Q Z,CAI D,ZHANG D,et al. Topic modeling with network regularizations[C]// Proceedings of the 17th International Conference on World Wide Web. 2008:101-110.

[8] PEROZZI B,AL-RFOU' R,SKIENA S. DeepWalk:online learning of social representations[C]// Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2014:701-710.

[9] TANG J,QU M,WANG M,et al. LINE:Large-scale Information Network Embedding[C]// Proceedings of the 24th International Conference on World Wide Web. 2015:1067-1077.

[10] GROVER A,LESKOVEC J. node2vec:Scalable Feature Learning for Networks[C]// Proceedings of International Conference on Knowledge Discovery & Data Mining(KDD). 2016:855-864.

[11] TU C C,LIU H,LIU Z Y,et al. CANE:Context-Aware Network Embedding for Relation Modeling[C]// Meeting of the Association for Computational Linguistics. 2017:1722-1731.

[12] PAN S R,WU J,ZHU X Q,et al. Tri-Party Deep Network Representation[C]// Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence. 2016:1895-1901.

[13] HUANG X,LI J D,HU X. Label Informed Attributed Network Embedding[C]// Proceedings of the Tenth ACM International Conference on Web Search and Data Mining. 2017:731-739.

[14] YANG C,LIU Z Y,ZHAO D L,et al. Network Representation Learning with Rich Text Information[C]// International Conference on Artificial Intelligence. AAAI Press,2015:2111-2117.

[15] NATARAJAN N,DHILLON I. Inductive matrix completion for predicting gene-disease associations[J]. Bioinformatics,2014,30(12):60-68.

[16] HOU C P,NIE F P,ZHANG C S,et al. Multiple rank multi-linear SVM for matrix data classification[J]. Pattern Recognition,2014,47(1):454-469.