软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2017,28(2):185−202 [doi: 10.13328/j.cnki.jos.004937]
©中国科学院软件研究所版权所有.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

求解随机时变背包问题的精确算法与进化算法*

贺毅朝[1,4], 王熙照[2], 李文斌[1], 赵书良[3,4]

[1](河北地质大学 信息工程学院,河北 石家庄 050031)

[2](深圳大学 计算机与软件学院,广东 深圳 518060)

[3](河北师范大学 数学与信息科学学院,河北 石家庄 050024)

[4](河北师范大学 软件学院,河北 石家庄 050024)

通讯作者: 贺毅朝, E-mail: heyichao@sjzue.edu.cn

摘 要: 随机时变背包问题(randomized time-varying knapsack problem,简称 RTVKP)是一种动态背包问题,也是一种动态组合优化问题,目前其求解算法主要是动态规划的精确算法、近似算法和遗传算法.首先,利用动态规划提出了一种求解 RTVKP 问题的精确算法,对算法时间复杂度的比较结果表明,它比已有的精确算法更适于求解背包载重较大的一类 RTVKP 实例.然后,分别基于差分演化和粒子群优化与贪心修正策略相结合,提出了求解 RTVKP 问题的两种进化算法.对 5 个 RTVKP 实例的数值计算结果比较表明,精确算法一般不宜求解大规模的 RTVKP 实例,而基于差分演化、粒子群优化和遗传算法与贪心修正策略相结合的进化算法却不受实例规模与数据大小的影响,对于振荡频率大且具有较大数据的大规模 RTVKP 实例均能求得一个极好的近似解.

关键词: 动态规划;时间复杂度;差分演化;粒子群优化;修复方法

中图法分类号: TP301

中文引用格式: 贺毅朝,王熙照,李文斌,赵书良.求解随机时变背包问题的精确算法与进化算法.软件学报,2017,28(2):185−202. http://www.jos.org.cn/1000-9825/4937.htm

英文引用格式: He YC, Wang XZ, Li WB, Zhao SL. Exact algorithms and evolutionary algorithms for randomized time-varying knapsack problem. Ruan Jian Xue Bao/Journal of Software, 2017,28(2):185−202 (in Chinese). http://www.jos.org.cn/1000-9825/4937.htm

Exact Algorithms and Evolutionary Algorithms for Randomized Time-Varying Knapsack Problem

HE Yi-Chao[1,4], WANG Xi-Zhao[2], LI Wen-Bin[1], ZHAO Shu-Liang[3,4]

[1](College of Information and Engineering, Hebei GEO University, Shijiazhuang 050031, China)

[2](College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China)

[3](College of Mathematics and Information Science, Hebei Normal University, Shijiazhuang 050024, China)

[4](Software College, Hebei Normal University, Shijiazhuang 050024, China)

Abstract: The randomized time-varying knapsack problem (RTVKP) is both a kind of dynamic knapsack problem and a kind of dynamic combinational optimization problem. Currently, the leading algorithms for solving RTVKP include the exact algorithm base on dynamic programming, approximation algorithm base on greedy-choice strategy and evolutionary algorithm base on genetic algorithm. First, in this paper, an exact algorithm base on dynamic programming to solve RTVKP is presented, along with comparison of its time complexity with the existing exact algorithms. Results show that the proposed algorithm is more suitable to solve RTVKP whose profit is larger. Then, the greedy correction and optimization strategy is combined with differential evolution and particle swarm optimization

* 基金项目: 国家自然科学基金(71371063, 61170040)
Foundation item: National Natural Science Foundation of China (71371063, 61170040)
收稿时间: 2014-08-04; 修改时间: 2015-01-21; 采用时间: 2015-10-30; jos 在线出版时间: 2015-11-18
CNKI 网络优先出版: 2015-11-18 14:58:47, http://www.cnki.net/kcms/detail/11.2560.TP.20151118.1458.002.html

© 中国科学院软件研究所 http://www.jos.org.cn

respectively to solve RTVKP. The numerical results on 5 instances of RTVKP show that the evolutionary algorithms which combine the differential evolution, particle swarm optimization and genetic algorithm with Greedy correction and optimization strategy respectively are more suitable to solve the hard RTVKP whose scale and oscillation frequency are larger while having bigger data.

Key words: dynamic programming; time complexity; differential evolution; particle swarm optimization; repair approach

背包问题(knapsack problem,简称 KP)[1]是计算机科学中的一类重要的 NPC 问题,也是一类经典的组合优化问题,在投资决策与资源分配等方面具有重要的应用背景[2,3].0-1 背包问题(0-1 knapsack problem,简称 0-1KP)[2]是最典型的 KP 问题,其一般描述为:从 n 个具有价值与重量的物品中选择若干个装入一个具有载重限制的背包,如何选择物品使得装入物品的重量之和在不超过背包载重前提下价值之和达到最大.

设第 $i(1\leqslant i\leqslant n)$个物品的价值与重量分别为 v_i 与 w_i,背包载重为 C,其中,v_i,w_i 与 C 均为正整数;令 $Y=(y_1,y_2,\ldots,y_n)\in\{0,1\}^n$ 表示 0-1KP 的一个可行解,当第 i 个物品被装入背包时 $y_i=1$,否则 $y_i=0$,则 0-1KP 的数学模型为:

$$\max f(Y)=\max\sum_{i=1}^{n}v_iy_i \tag{1}$$

$$\text{s.t.}\quad \sum_{i=1}^{n}w_iy_i\leqslant C \tag{2}$$

动态背包问题(dynamic knapsack problem,简称 DKP)[4,5]是 0-1KP 的一种动态扩展形式.在 DKP 中,物品的价值、重量或背包载重不再固定不变,而是随着时间的推移可能变大或缩小,因此 DKP 是一种比 0-1KP 更不易求解的动态组合优化问题.

Goldberg 等人[4]首先提出了背包载重在两个固定值之间振荡变化的 DKP:时变背包问题(time-varying knapsack problem,简称 TVKP),并且他们利用二倍体遗传算法(genetic algorithm,简称 GA)求解 TVKP 问题.随后,Hadad 等人[6]改进了 Goldberg 等人的方法,提出了求解 TVKP 的多倍体 GA,并指出其对于变化频率较大的 TVKP 多倍体方法更具有优势.Yang [7]利用原对偶遗传算法(primal-dual genetic algorithm,简称 PDGA)求解 TVKP 问题,计算结果表明其求解效果优于基本 GA.Lewis 等人[8]比较了几种多倍体方法在求解 TVKP 时的优劣,指出当背包载重在 $k(k\geqslant 3)$个固定值之间振荡变化时多倍体方法的优势并不非常明显.周传华等人[9]提出了一种带动态小生境的自组织学习算法(DNSLA),并将 DNSLA 用于求解多个规模为 100 的 TVKP 实例.贺毅朝等人[10,11]利用动态规划(dynamic programming,简称 DP)求解背包载重在给定区间内随机振荡变化的 TVKP 问题,提出了求解它的两种精确算法.李宁等人[12]则利用具有混合编码的和声搜索算法求解背包载重在给定区间内随机振荡变化的 TVKP 问题.

He 等人[13]将 TVKP 推广为随机时变背包问题(randomized time-varying knapsack problem,简称 RTVKP),并分别利用 DP、贪心算法和 GA 给出了求解 RTVKP 问题的精确算法、近似算法和进化算法,初步讨论了求解动态组合优化问题的近似算法优劣的评价标准.由于 0-1KP 是 RTVKP 的特例,因此 RTVKP 也是一个 NPC 问题.此外,随着时间的推移,RTVKP 中物品的价值、重量以及背包载重均可能变大或减小,故而 RTVKP 是比 TVKP 的求解难度更大的 DKP.在本文中,首先利用 DP 提出一种求解 RTVKP 的新精确算法,然后分别基于差分演化和粒子群优化给出求解 RTVKP 的两种有效进化算法.

本文第 1 节给出 RTVKP 的定义与数学模型,并简要分析其特性.第 2 节介绍利用第二动态规划法求解 0-1KP 的基本算法,利用它们提出一种求解 RTVKP 的精确算法,并比较其与已有精确算法的时间复杂度.第 3 节首先给出基于进化算法求解 RTVKP 时处理不可行解的有效算法 GOA[13],然后分别利用差分演化和粒子群优化与 GOA 相结合,提出求解 RTVKP 问题的两种进化算法.第 4 节通过对 5 个较大规模 RTVKP 实例的仿真计算,验证进化算法求解 RTVKP 问题的高效性与实用性.最后,总结全文,并提出今后进一步的研究思路.

1 RTVKP 问题及其数学模型

所谓随机时变背包问题(RTVKP)[13],即给定 n 个物品和一个背包,每个物品均具有价值与重量两种属性,背包具有载重限制,并且物品的价值和重量以及背包载重均可以随着时间的推移而动态随机振荡变化.如何在每一变化间隔周期内选择物品装入背包,使得在不超过背包当前载重的前提下装入物品的价值之和达到最大.

© 中国科学院软件研究所 http://www.jos.org.cn

设 RTVKP 中 n 个物品的初始价值集与初始重量集分别为 $V_0=\{v_{01},v_{02},\ldots,v_{0n}\}$ 和 $W_0=\{w_{01},w_{02},\ldots,w_{0n}\}$,$v_{0j}\in[A_v,B_v]$,$w_{0j}\in[A_w,B_w]$$(1\leqslant j\leqslant n)$,背包初始载重为 $C_0\in[A_c,B_c]$,其中 A_v,B_v,A_w,B_w,A_c 和 B_c 均为正整数,且 $A_v<B_v,A_w<B_w,B_w<A_c<B_c<nA_w$.令 $T_i(i\geqslant 1)$表示物品的价值、重量或背包载重的第 i–1 次随机振荡变化与第 i 次随机振荡变化之间的时间间隔(即在此间隔内物品的价值、重量和背包载重都是固定不变的),称为第 i 次随机变化周期(单位为 s).记第 $i(i\geqslant 1)$次随机变化后 n 个物品的价值集与重量集分别为 $V_i=\{v_{i1},v_{i2},\ldots,v_{in}\}$ 和 $W_i=\{w_{i1},w_{i2},\ldots,w_{in}\}$,背包载重为 $C_i\in[A_c,B_c]$,其中 $v_{ij}\in[A_v,B_v]$,$w_{ij}\in[A_w,B_w]$$(1\leqslant j\leqslant n)$,并且满足$|(V_i\cup W_i)-(V_{i-1}\cup W_{i-1})|\leqslant Threshold$(一般取 $Thrshold\leqslant 3\sqrt{n}$),$|(V_i\cup W_i)-(V_{i-1}\cup W_{i-1})|+|C_i-C_{i-1}|>0$.记 $\mathrm{RTVKP}_{i-1}(n,C_{i-1},V_{i-1},W_{i-1})(i\geqslant 1)$表示在随机变化周期 T_i 中以 V_{i-1} 为价值集,W_{i-1} 为重量集,C_{i-1} 为背包载重所对应的 0-1KP 子问题(进一步简记为 RTVKP_{i-1}),则 RTVKP 即为依次求解时间间隔序列$\{T_i\}_{i\geqslant 1}$上的一系列 $\mathrm{RTVKP}_{i-1}(i\geqslant 1)$子问题的最优值与最优解.

令 $Y_i=(y_{i1},y_{i2},\ldots,y_{in})\in\{0,1\}^n$ 表示 $\mathrm{RTVKP}_i(i\geqslant 0)$的可行解,则此时第 $j(1\leqslant j\leqslant n)$个物品装入载重为 C_i 的背包时 $y_{ij}=1$,否则 $y_{ij}=0$.于是,在 T_{i+1} 的时间限制下,RTVKP 的数学模型描述如下:

$$\max f(Y_i)=\max\sum_{j=1}^{n}y_{ij}v_{ij} \tag{3}$$

$$\text{s.t.}\quad \sum_{j=1}^{n}y_{ij}w_{ij}\leqslant C_i \tag{4}$$

其中,$i\geqslant 0$,$1\leqslant j\leqslant n$;$V_i=\{v_{i1},v_{i2},\ldots,v_{in}\}$ 和 $W_i=\{w_{i1},w_{i2},\ldots,w_{in}\}$分别为 RTVKP_i 对应的价值集和重量集,C_i 为背包载重.

当 RTVKP 的物品价值集和重量集均不发生变化时,即$(V_i\cup W_i)-(V_{i-1}\cup W_{i-1})=\varnothing$时,RTVKP 退化为 TVKP.显然,RTVKP 是由一系列具有部分共同数据的 0-1KP 子问题构成的,即子问题 RTVKP_i 与 $\mathrm{RTVKP}_{i+1}(i\geqslant 0)$的大部分物品的价值与重量是相同的,只有少部分物品的价值、重量或背包载重是相异的.正是由于 RTVKP 的子问题之间的这种共有数据特征,使得能够利用 DP 对其进行求解.

令 $\mathbf{Min}\boldsymbol{T}=\min\{T_i|T_i$ 为 RTVKP 的第 i 次随机变化周期,$i\geqslant 1\}$,则 RTVKP 实例的 $\mathbf{Min}\boldsymbol{T}$ 值越小,要求算法适应其振荡变化的能力就越高,实例就越不容易被成功求解.此外,当 RTVKP 问题的随机变化周期 T_i 恒为常数时,称其为具有固定变化周期的 RTVKP,并简记为 fixRTVKP.

2 基于动态规划法求解 RTVKP 问题

动态规划(DP)[11,14−16]是一种重要的算法设计方法,适于求解具有最优子结构性质的最优化问题.在利用 DP 求解问题时,刻画子问题的最优解的结构特征,并依此建立问题与子问题最优值之间的递推公式是算法设计的关键所在;一旦建立了递推公式,就可以根据它按照自底向上的方式逐一求解各子问题的最优值,最终求得原问题的最优值(这一过程通常利用填表实现);然后,根据求得的最优值和递推公式,按照自顶向下的方式逐步确定问题的一个最优解(这一过程通常是按照与填表相反的顺序查表实现).

下面首先给出求解 0-1KP 的第二动态规划法[11,16],然后利用它提出一种求解 RTVKP 问题的具有伪多项式时间的新精确算法.

设 0-1KP 问题的价值集与重量集分别为 $V=\{v_1,v_2,\ldots,v_n\}$ 和 $W=\{w_1,w_2,\ldots,w_n\}$,背包载重为 C.为了建立 0-1KP 最优值的递推公式,令 U_{ik} 表示从前 i 个物品中选择物品装入背包,当所选择物品的价值之和等于 k 时重量之和所能达到的最小值,其中,$1\leqslant i\leqslant n$,$0\leqslant k\leqslant P$ 且 $P=\sum_{j=1}^{n}v_j$;如果前 i 个物品中不存在的价值之和等于 k 的物品子集,则令 $U_{ik}=\infty$,于是有递推公式

$$U_{ik}=\begin{cases}\min\{U_{i-1,k},U_{i-1,k-v_i}+w_i\}, & \text{if } v_i\leqslant k,\ i\geqslant 2\\ U_{i-1,k}, & \text{if } v_i>k,\ i\geqslant 2\end{cases} \tag{5}$$

初始条件为 $U_{i0}=0(1\leqslant i\leqslant n)$,$U_{1k}=\infty(1\leqslant k\leqslant P,k\neq v_1)$且 $U_{1v_1}=w_1$.由式(5)可知,U_{ik} 的计算与背包载重无关,但与所有物品的价值之和是相关的.计算出所有 $U_{nk}(1\leqslant k\leqslant P)$之后,即可由 $OPT=\max_{1\leqslant k\leqslant P}\{k\,|\,U_{nk}\leqslant C\}$ 求得 0-1KP 的最优值.

设 $U_{ik}(1\leqslant i\leqslant n,0\leqslant k\leqslant P)$存于二维数组 $U[1\ldots n,0\ldots P]$中,其中 $P=\sum_{j=1}^{n}v_j$;令 P_0,low 和 $high$ 均为整型变量,并且 $1\leqslant low\leqslant high\leqslant n$,$P_0\leqslant P$;记 T 为给定的求解周期,函数 $SystemTime(\ )$用于获取系统当前时间,于是在周期 T

© 中国科学院软件研究所 http://www.jos.org.cn

内求 $U_{ik}(1\leqslant i\leqslant n,0\leqslant k\leqslant P)$的算法 OPTfor0-1KP 描述如下.

算法 1. OPTfor0-1KP.

输入:集合 V 和 W;参数 $C,T,n,low,high,P_0$ 和 P.

输出:二维数组 $U[low...high,P_0\ldots P]$.

1. $time\leftarrow SystemTime(\ )$;
2. **if** $P_0=0$ **then**
3. **for** $i=low$ **to** $high$ **do** $U[i,0]\leftarrow 0$;
4. **end if**
5. **if** $low=1$ **then**
6. **for** $k=1$ **to** P **do** $U[1,k]\leftarrow\infty$;
7. $U[1,v_1]\leftarrow w_1$;
8. **end if**
9. **for** $i=low$ **to** $high$ **do**
10. **for** $k=P_0$ **to** P **do**
11. $U[i,k]\leftarrow U[i-1,k]$;
12. **if** $v_i\leqslant k$ **then** $U[i,k]\leftarrow \min\{U[i-1,k],U[i-1,k-v_i]+w_i\}$;
13. **if** $SystemTime(\ )-time\geqslant T$ and ($k<P$ or $i<high$) **then return** ("Solving failure");
14. **end for**
15. **end for**
16. **return** ($U[low...high,P_0\ldots P]$).

算法 1 的时间复杂度为 $O((high-low+1)(P-P_0+1))$.注意到,当 $P>>P_0$ 时,$P-P_0+1$ 是一个大整数,$\log_2(P-P_0+1)$为其输入规模,因此算法 1 是一种伪多项式时间精确算法.在求 0-1KP 时,首先利用算法 1 计算 $U[1\ldots n,0\ldots P]$,其调用方式为 $OPTfor0\text{-}1KP(n,V,W,C,1,n,0,P,T)$,然后由 $OPT=\max\limits_{1\leqslant k\leqslant P}\{k\mid U[n,k]\leqslant C\}$ 即可求得最优值.

在算法 1 的基础上,求解 0-1KP 最优解的算法 SOLfor0-1KP 描述如下.

算法 2. SOLfor0-1KP.

输入:集合 V 和 W;参数 n,C,OPT;二维数组 $U[1\ldots n,0\ldots P]$.

输出:0-1KP 问题的最优解 $Y=(y_1,y_2,\ldots,y_n)\in\{0,1\}^n$.

1. $i\leftarrow n$; $k\leftarrow OPT$;
2. **for** $i=1$ **to** n **do** $y_i\leftarrow 0$;
3. **while** ($i>1$ and $k>0$)**do**
4. **if** $U[i,k]=U[i-1,k-v_i]+w_i$ **then** $y_i\leftarrow 1$ and $k\leftarrow k-v_i$;
5. $i\leftarrow i-1$;
6. **end while**
7. **if** ($i=1$ and $U[1,k]=w_1$) **then** $y_1\leftarrow 1$;
8. **return** (Y).

算法 2 输出 0-1KP 的一个最优解 Y,其算法时间复杂度为 $\Theta(n)$;在算法 2 的输入参数中,OPT 是最优解 Y 对应的最优值.对于 0-1KP,利用算法 2 求其最优解的调用方式为 $SOLfor0\text{-}1KP(n,V,W,C,OPT,U[1\ldots n,0\ldots P])$.

下面首先利用算法 1 和算法 2 基于动态规划法阐述求解 RTVKP 问题的算法(记为 DPfRTVKP)设计思想,然后给出其算法伪代码描述.

初始时,调用算法 1 和算法 2 在随机变化周期 T_1 内直接求解子问题 $RTVKP_0(n,C_0,V_0,W_0)$.

假设子问题 $RTVKP_i(n,C_i,V_i,W_i)(i\geqslant 0)$已经求解完毕,并且其某些物品的价值和重量或背包载重发生变化得到后继子问题 $RTVKP_{i+1}(n,C_{i+1},V_{i+1},W_{i+1})$,则按照如下的两种情形对 $RTVKP_{i+1}$ 进行求解.

© 中国科学院软件研究所 http://www.jos.org.cn

(1) 当$(V_{i+1}\cup W_{i+1})-(V_i\cup W_i)=\varnothing$时,说明 RTVKP$_i$ 随机变化为 RTVKP$_{i+1}$ 时仅有背包载重发生了改变,物品的价值与重量均未改变,此时直接利用 $OPT=\max_{1\leqslant k\leqslant P}\{k\mid U_{nk}\leqslant C_{i+1}\}$ 和算法 2 求解即可.

(2) 当$(V_{i+1}\cup W_{i+1})-(V_i\cup W_i)\neq\varnothing$时,说明 RTVKP$_i$ 随机变化为 RTVKP$_{i+1}$ 时至少有 1 个物品的价值或重量发生了改变,于是令 $P_i=\sum_{j=1}^{n}v_{ij}$, $P_{i+1}=\sum_{j=1}^{n}v_{i+1,j}$, 并设 *low* 为集合$(V_{i+1}\cup W_{i+1})-(V_i\cup W_i)$中物品的最小下标,则根据 P_i 与 P_{i+1} 的大小情况利用算法 1 分别计算如下:

(2.1) 如果 $P_i\geqslant P_{i+1}$,则求解 RTVKP$_i$ 时已经计算的 $U_{ik}(1\leqslant i\leqslant low-1,0\leqslant k\leqslant P_{i+1})$对于 RTVKP$_{i+1}$ 仍适用,此时只需利用算法 1 重新计算 $U_{ik}(low\leqslant i\leqslant n,0\leqslant k\leqslant P_{i+1})$即可;

(2.2) 如果 $P_i<P_{i+1}$,则求解 RTVKP$_i$ 时已经计算的 $U_{ik}(1\leqslant i\leqslant low-1,0\leqslant k\leqslant P_i)$对于 RTVKP$_{i+1}$ 仍适用,此时只需利用算法 1 分别计算 $U_{ik}(1\leqslant i\leqslant low-1,P_i+1\leqslant k\leqslant P_{i+1})$和 $U_{ik}(low\leqslant i\leqslant n,1\leqslant k\leqslant P_{i+1})$即可.

在按照上述方法计算出 $U_{ik}(1\leqslant i\leqslant n,0\leqslant k\leqslant P_{i+1})$之后,利用 $OPT=\max_{1\leqslant k\leqslant P}\{k\mid U_{nk}\leqslant C_{i+1}\}$ 求得 RTVKP$_{i+1}$ 的最优值,然后再调用算法 2 求得其最优解.

令 *m* 为 RTVKP 的随机振荡变化次数,则 RTVKP 可视为由 *m* 个 0-1KP 子问题构成.令 $V=\{v_1,v_2,\ldots,v_n\}$与 $W=\{w_1,w_2,\ldots,w_n\}$分别表示子问题的价值集和重量集,*C* 表示背包载重.设函数 *CheckVary*(V_0,W_0,C_0)的作用是在 V_0,W_0 和 C_0 的基础上,在区间$[A_v,B_v]$,$[A_w,B_w]$和$[A_c,B_c]$上随机产生新的价值集 *V*、重量集 *W* 和背包载重 *C*,函数 *GetPeriod*( )用于产生新的随机变化周期 *T*,函数 *FindIndex*$((V\cup W)-(V_0\cup W_0))$用于确定集合$(V\cup W)-(V_0\cup W_0)$中物品的最小下标,于是 DPfRTVKP 的算法伪代码描述如下.

算法 3. DPfRTVKP.

输入:初始价值集 $V_0=\{v_{01},v_{02},\ldots,v_{0n}\}$;初始重量集 $W_0=\{w_{01},w_{02},\ldots,w_{0n}\}$;初始背包载重 C_0;参数 *n*,*m*,*Threshold*, A_v,B_v,A_w,B_w,A_c 和 B_c.

输出:RTVKP 的 *m* 个子问题的最优值与最优解.

1. $V\leftarrow V_0$; $W\leftarrow W_0$; $C\leftarrow C_0$;
2. $P_0\leftarrow 0$; $P\leftarrow\sum_{j=1}^{n}v_j$; *low*$\leftarrow$1; *high*$\leftarrow n$; *k*$\leftarrow$0;
3. *T*$\leftarrow$*GetPeriod*(); *time*$\leftarrow$*SystemTime*();
4. $(U[low\ldots high,P_0\ldots P])\leftarrow$***OPTfor*0-1*KP***$(n,V,W,C,low,high,P_0,P,T)$;
5. **while** $(k<m)$ **do**
6. **if** $(V\cup W)-(V_0\cup W_0)\neq\varnothing$ **then**
7. **if** $P_0<P$ **then**
8. $(U[1\ldots low-1,P_0+1\ldots P])\leftarrow$***OPTfor*0-1*KP***$(n,V,W,C,1,low-1,P_0+1,P,T)$;
9. $T\leftarrow T-$*SystemTime*()+*time*;
10. **end if**
11. $(U[low\ldots n,0\ldots P])\leftarrow$***OPTfor*0-1*KP***$(n,V,W,C,low,n,0,P,T)$;
12. **endif**
13. $OPT\leftarrow\max_{1\leqslant j\leqslant P}\{j\mid U[n,j]\leqslant C\}$;
14. $Y\leftarrow$***SOLfor*0-1*KP***$(n,V,W,C,OPT,U[1\ldots n,0,\ldots P])$;
15. **return** (OPT,Y);
16. **while** (*SystemTime*( )$-$*time*$<T$) **do** *SystemTime*( );
17. **if** $(k<m-1)$ **then**
18. $V_0\leftarrow V$; $W_0\leftarrow W$; $C_0\leftarrow C$; $P_0\leftarrow P$; *T*$\leftarrow$*GetPeriod*( );
19. $(V,W,C)\leftarrow$*CheckVary*(V_0,W_0,C_0); $P\leftarrow\sum_{j=1}^{n}v_j$;
20. *low*$\leftarrow$*FindIndex*$((V\cup W)-(V_0\cup W_0))$; *time*$\leftarrow$*SystemTime*( );
21. **end if**

© 中国科学院软件研究所 http://www.jos.org.cn

22. $k \leftarrow k+1$;

23. **end while**

24. **return** ("Success to solve").

注意到 RTVKP 的规模由物品个数 n 和子问题数 m 确定,记 $P_{\max}=\max(\{|P_i-P_{i-1}||1\leqslant i\leqslant m-1\}\cup\{P_0\})$,其中,$P_i=\sum_{j=1}^{n}v_{ij}$ $(0\leqslant i\leqslant m-1)$;记 l_i 为$(V_i\cup W_i)$–$(V_{i-1}\cup W_{i-1})$中物品的最小下标,则算法 DPfRTVKP 的时间复杂度为 $O(nP_0)+\sum_{i=1}^{m-1}((n-l_i+1)P_i+(l_i-1)\times|P_i-P_{i-1}|)=O(nmP_{\max})$.

由于 $P_{\max}$ 往往是一个大整数,$\log_2 P_{\max}$ 为其输入规模,所以算法 3 是一种具有伪多项式时间的精确算法.显然,$O(nP_i)<T_{i+1}(0\leqslant i\leqslant m-1)$是保证 DPfRTVKP 能够成功求解 RTVKP 中每一个子问题的充分条件,因此 $O(nP_{\max})<\mathbf{Min}\boldsymbol{T}$ 是保证 DPfRTVKP 能够成功求解 RTVKP 问题的一个充分条件.

类似地,令 $C_{\max}=\max(\{|C_i-C_{i-1}||1\leqslant i\leqslant m-1\}\cup\{C_0\})$,$C_i(0\leqslant i\leqslant m-1)$为 RTVKP_i 的背包载重,则算法 DPMfRDKP[13] 的时间复杂度为 $O(nC_0)+\sum_{i=1}^{m-1}((n-l_i+1)C_i+(l_i-1)\times|C_i-C_{i-1}|)=O(nmC_{\max})$.

比较 DPfRTVKP 与 DPMfRDKP 的时间复杂度以及成功求解 RTVKP 问题的充分条件易知:在保证算法能够成功求解 RTVKP 问题的前提下,当 $O(nmP_{\max})<O(nmC_{\max})$时,算法 DPfRTVKP 的求解速度快;当 $O(nmC_{\max})<O(nmP_{\max})$时,算法 DPMfRDKP 的求解速度快.于是有如下 r 结论.

结论 1. 当 $P_{\max}<\min\{C_i|0\leqslant i\leqslant m-1\}$时,DPfRTVKP 更适于求解 RTVKP 问题;当 $C_{\max}<\min\{P_i|0\leqslant i\leqslant m-1\}$时,DPMfRDKP 则更适于求解 RTVKP 问题.

3 基于进化算法求解 RTVKP 问题

进化算法(evolutionary algorithm,简称 EA)[17,18]是一类群智能启发式算法,常见的有遗传算法(genetic algorithm,简称 GA)[18]、差分演化(differential evolution,简称 DE)[19–21]、蚁群优化(ant colony optimization,简称 ACO)[22]、粒子群优化(particle swarm optimization,简称 PSO)[23,24]、混合蛙跳算法(shuffled frog-leaping algorithm,简称 SFLA)[25]与和声搜索算法(harmony search algorithm,简称 HSA)[12,26]等.目前 EAs 已被广泛应用于求解各种组合优化问题,如 0-1KP 问题、可满足性问题(satisfiability problem,简称 SAT)[27]和旅行商问题(traveling salesman problem,简称 TSP)[28]等,其中利用 GA,PSO,DE 和 HSA 及其改进算法求解 0-1KP 和 TVKP 已被众多学者所研究[4,6–9,12,21,29,30].在本节中,我们首先给出利用 EAs 求解 RTVKP 实例时处理不可行解的一种有效方法,然后借鉴文献[13]中的算法设计思想,基于 HBDE[21]和 DS_BPSO[24]分别给出求解 RTVKP 的两种进化算法.

3.1 一种处理不可行解的有效方法

RTVKP 是一个约束优化问题,在利用 EAs 求解时需要解决的一个关键问题是如何保证个体编码满足约束条件.通常将不满足约束条件的个体编码对应的解称为不可行解.对于约束优化问题,处理不可行解的常见方法[31–33]有:罚函数法(penalty function approach)、修复法(repair approach)、纯正法(purist approach)、分离法(separatist approach)和混合方法(hybrid approach).罚函数法通过在目标函数中引入一个惩罚项,将约束优化问题转化为非约束优化问题.修复法是一种将不可行解按照某种策略修正为可行解的处理方法.纯正法是在进化过程中通过拒绝对应不可行解的个体来满足约束条件.分离法是一种将目标函数和约束条件分开进行处理的方法.混合法是指同时使用以上至少两种方法的综合处理方法.

对于一般的约束优化问题,以上方法各有利弊;但对于 0-1KP,Michalewicz 等人[33]经过研究指出修复法是最为适宜的,并且基于贪心策略给出了一种修复不可行解的有效方法.注意到,RTVKP 本质上是由一系列具有部分共同数据的 0-1KP 子问题构成,因此在利用 EAs 求解时采用修复法处理不可行解是最佳选择.正是基于以上原因,He 等人[13]利用修复法与遗传算法相结合求解 RTVKP 问题,并在 Michalewicz 等人[33]所提出方法的基础上引入优化策略,给出了一种效果更佳的修复方法:贪心优化算法(greedy optimization algorithm,简称 GOA)[13].下面

© 中国科学院软件研究所 http://www.jos.org.cn

我们给出 GOA 的伪代码描述.

设 $Y=(y_1,y_2,\ldots,y_n)\in\{0,1\}^n$ 为 EAs 的某个体对应 0-1KP 实例的 0-1 解向量,注意,解向量 Y 不一定为 0-1KP 实例的可行解.令 $H[1\ldots n]$为一个二维数组,用于存放利用快速排序算法(QuickSort)[14,15]按照物品的价值与重量的比值大小对物品进行排序后物品的下标,则 GOA 的算法伪代码描述如下.

算法 4. GOA.

输入:0-1KP 实例的维数 n;价值集 $V=\{v_1,v_2,\ldots,v_n\}$;重量集 $W=\{w_1,w_2,\ldots,w_n\}$;背包载重 C;个体 $Y=(y_1,y_2,\ldots,y_n)$;二维数组 $H[1\ldots n]$.

输出:修复与优化后的可行解 $Y=(y_1,y_2,\ldots,y_n)$及其目标函数值 $f(Y)$.

1. $fweight\leftarrow 0$; $fvalue\leftarrow 0$;
2. **for** $i=1$ **to** n **do** $fweight\leftarrow fweight+y_i\times w_i$;
3. **if** $fweight>C$ **then**
4. $temp\leftarrow 0$;
5. **for** $i=1$ **to** n **do**
6. $temp\leftarrow temp+y_{H[i]}\times w_{H[i]}$;
7. **if** $temp>C$ **then** $temp\leftarrow temp-y_{H[i]}\times w_{H[i]}$ and $y_{H[i]}\leftarrow 0$;
8. **end for**
9. $fweight\leftarrow temp$;
10. **end if**
11. **for** $i=1$ **to** n **do**
12. **if** ($y_{H[i]}=0$ and $fweight+w_{H[i]}\leqslant C$) **then** $y_{H[i]}\leftarrow 1$ and $fweight\leftarrow fweight+w_{H[i]}$;
13. **end for**
14. **for** $i=1$ **to** n **do** $fvalue\leftarrow fvalue+y_i\times v_i$;
15. **return** (Y,$fvalue$);

在算法 4 中,若个体 Y 是不可行解,则 Step 3~Step 10 用于对其进行修正,否则不起作用;Step 11~Step 13 用于对可行解(包括修复后的可行解)做进一步的优化,Step 14 则用于计算 Y 的目标函数值 $f(Y)$.最后,算法输出优化后的可行解 Y 及其适应度值 $f(Y)=fvalue$.显然,GOA 的时间复杂度为 $\Theta(n)$.

3.2 基于差分演化求解RTVKP问题

差分演化(DE)[19,20]是由 Rainer Storn 和 Kenneth Price 于 1996 年提出的一种具有极强全局搜索能力的 EAs,因其在第 1 届 IEEE 演化大赛中表现超群,引起了国内外学者的极大关注.目前,DE 已被广泛应用于求解许多领域中的数值优化问题和组合优化问题[19–21,34–37].为了利用 DE 求解问题的解可表示为 0-1 向量的组合优化问题,贺毅朝等人[21]提出了一种具有混合编码的二进制差分演化算法(HBDE),并利用它求解 0-1KP 和 SAT 问题.下面首先基于最大优化问题简介 HBDE 的基本原理,然后将其与 GOA 相结合,提出一种求解 RTVKP 问题的进化算法.

记 $\boldsymbol{P}(t)$为 HBDE 的第 $t(t\geqslant 0)$代种群,其第 i 个个体的混合编码为$(X_i(t),Y_i(t))$,其中 $X_i(t)=(x_{i1}(t),x_{i2}(t),\ldots,x_{in}(t))\in\prod_{j=1}^{n}[L_j,U_j]$, $Y_i(t)=(y_{i1}(t),y_{i2}(t),\ldots,y_{in}(t))\in\{0,1\}^n$ 为问题的一个潜在解,$1\leqslant i\leqslant N$,N 为种群规模,n 为问题的维数,L_j 和 U_j 均为实数且 $L_j<U_j(1\leqslant j\leqslant n)$.记 $\boldsymbol{P}(t)$中最优个体为$(U(t),B(t))$,其中,$U(t)=(u_1(t),u_2(t),\ldots,u_n(t))\in\prod_{j=1}^{n}[L_j,U_j]$, $B(t)=(b_1(t),b_2(t),\ldots,b_n(t))\in\{0,1\}^n$.又记 $\boldsymbol{Q}(t)$为 HBDE 的第 $t(t\geqslant 0)$代中间种群,其第 i 个中间个体的混合编码为$(Z_i(t),E_i(t))$,$Z_i(t)=(z_{i1}(t),z_{i2}(t),\ldots,z_{in}(t))\in\prod_{j=1}^{n}[L_j,U_j]$, $E_i(t)=(e_{i1}(t),e_{i2}(t),\ldots,e_{in}(t))\in\{0,1\}^n$,于是,由 $\boldsymbol{P}(t)$产生 $\boldsymbol{Q}(t)$的差异算子定义为

$$z_{ij}(t)=\begin{cases}x_{p_1j}(t)+\alpha(x_{p_2j}(t)-x_{p_3j}(t)), & \text{if } r\leqslant CR \text{ or } j=R(i)\\ x_{ij}(t), & \text{otherwise}\end{cases} \tag{6}$$

© 中国科学院软件研究所 http://www.jos.org.cn

$$e_{ij}(t)=\begin{cases}1, & sig(z_{ij}(t))\geqslant 0.5\\ 0, & \text{otherwise}\end{cases} \tag{7}$$

其中,$1\leqslant i\leqslant N$,$1\leqslant j\leqslant n$;$(X_{p1}(t),Y_{p1}(t))$,$(X_{p2}(t),Y_{p2}(t))$,$(X_{p3}(t),Y_{p3}(t))$分别是 $\boldsymbol{P}(t)$中 3 种不同于$(X_i(t),Y_i(t))$的不同个体.$0<\alpha<1$ 称为缩放因子,$CR\in(0,1)$称为变异因子,r 是(0,1)上的一个随机实数,$R(i)$是$[1,n]$上的一个随机正整数,$sig(x)=1/(1+e^{-x})$是模糊函数.

在 HBDE 的差异算子中,混合编码个体$(X(t),Y(t))$的第 2 个分量 $Y(t)$并不参与产生中间个体$(Z(t),E(t))$的计算,而是用于表示第 1 个分量 $X(t)$对应于问题的潜在解.显然,这种编码方法可以使 HBDE 保持 DE 原有的进化方式,从而继承 DE 的全局搜索能力.

HBDE 的选择算子是一种贪心选择算子,即如果中间种群 $\boldsymbol{Q}(t)$中个体$(Z_i(t),E_i(t))$优于种群 $\boldsymbol{P}(t)$中的个体$(X_i(t),Y_i(t))$,则选择$(Z_i(t),E_i(t))$作为第 t+1 代种群 $\boldsymbol{P}(t+1)$中的第 i 个个体$(X_i(t+1),Y_i(t+1))$,否则选择$(X_i(t),Y_i(t))$,即

$$(X_i(t+1),Y_i(t+1))=\begin{cases}(Z_i(t),E_i(t)), & \text{if } f(E_i(t))>f(Y_i(t))\\ (X_i(t),Y_i(t)), & \text{othewise}\end{cases} \tag{8}$$

其中,$1\leqslant i\leqslant N$,$f(Y)$为 Y 对应的目标函数值(一般可作为个体(X,Y)的适应度).

对于最大优化问题,记 $\boldsymbol{P}(t)$中最优个体$(U(t),B(t))$为适应度最大的个体,即$(U(t),B(t))$是 $\boldsymbol{P}(t)$中满足 $f(B(t))\leqslant f(Y_i(t))(1\leqslant i\leqslant N)$的个体.由于选择算子是贪心方式的,因此$(U(t),B(t))$必为 HBDE 前 t 代进化获得的全局最优个体.

令 m 为 RTVKP 的随机振荡变化次数,$V=\{v_1,v_2,\ldots,v_n\}$与 $W=\{w_1,w_2,\ldots,w_n\}$分别为子问题的价值集和重量集,C 为背包载重.记"$H[1\ldots n]\leftarrow QuickSort(\{v_i/w_i|v_i\in V,w_i\in W,1\leqslant i\leqslant n\})$"表示利用快速排序算法(QuickSort)[14,15]按照物品价值与重量的比值对物品进行排序后将物品下标依次存储于数组 $H[1\ldots n]$中.函数 $SystemTime(\,)$,$CheckVary(V,W,C)$,$GetPeriod(\,)$和 $FindIndex((V\cup W)-(V_0\cup W_0))$和 T 的含义同上,于是利用 HBDE 与 GOA 相结合求解 RTVKP 的进化算法(记为 HBDEfRTVKP)的伪代码描述如下.

算法 5. HBDEfRTVKP.

输入:初始价值集 $V_0=\{v_{01},v_{02},\ldots,v_{0n}\}$;初始重量集 $W_0=\{w_{01},w_{02},\ldots,w_{0n}\}$;初始背包载重 C_0;参数 $n,m,Threshold,A_v,B_v,A_w,B_w,A_c$ 和 B_c;种群规模 N,L_j 和 $U_j(1\leqslant j\leqslant n)$.

输出:RTVKP 的 m 个子问题的最优值和最优解.

```
1.  V←V0; W←W0; C←C0; k←0;
2.  T←GetPeriod( ); time←SystemTime( );
3.  while (k<m) do
4.      Generate the initial population P(0)={(Xi(0),Yi(0))|1≤i≤N} randomly;
5.      t←0;
6.      H[1…n]←QuickSort({vi/wi|vi∈V,wi∈W,1≤i≤n});
7.      for i=1 to N do (Yi(t),f(Yi(t)))←GOA(n,V,W,C,Yi(t),H[1…n]);
8.      Get (U(t),B(t)) from P(t) according to f(Yi(t)) (1≤i≤N);
9.      while (SystemTime( )−time<T) do
10.         for i=1 to N do
11.             for j=1 to n do
12.                 if (≤CR∨j=R(i)) then zij←yp1,j+α(yp2,j−yp3,j) else zij←yij;
13.                 if sig(zij)≥0.5 then eij←1 else eij←0;
14.             endfor
15.             if (SystemTime( )−time≥T) then goto 23;
16.             (Ei(t),f(Ei(t)))←GOA(n,V,W,C,Ei(t),H[1…n]);
17.             if f(Ei(t)>f(Yi(t)) then (Xi(t+1),Yi(t+1))←(Zi(t),Ei(t));
18.             else (Xi(t+1),Yi(t+1))←(Xi(t),Yi(t));
```

© 中国科学院软件研究所 http://www.jos.org.cn

19. **endfor**
20. $t\leftarrow t+1$;
21. Get $(U(t),B(t))$ from $\boldsymbol{P}(t)$ according to $f(Y_i(t))$ $(1\leqslant i\leqslant N)$;
22. **endwhile**
23. **output** $(f(B(t)),B(t))$;
24. **if** $(k<m-1)$ **then**
25. $V_0\leftarrow V$; $W_0\leftarrow W$; $C_0\leftarrow C$; $T\leftarrow GetPeriod(\ )$;
26. $(V,W,C)\leftarrow CheckVary(V_0,W_0,C_0)$; $time\leftarrow SystemTime(\ )$;
27. **end if**
28. $k\leftarrow k+1$;
29. **end while**
30. **return** ("Success to solve").

由于算法 5 是一种随机近似算法,只要时间允许它将一直保持进化搜索.因此,在 RTVKP 的每个随机变化周期内,算法 5 将一直保持求解状态,讨论其时间复杂度是没有意义的,故而下面重点分析它能够求得 RTVKP 问题的一个较好近似解应该满足的条件.注意到算法 5 中 Step 4 和 Step 7 的时间复杂度均为 $\Theta(nN)$,Step 6 的时间复杂度为 $\Theta(n\log n)$,Step 8 的时间复杂度为 $\Theta(n+N)$,而对于 RTVKP 的每一个子问题,只要算法 5 能够产生至少 1 代种群(即执行完 Step 4~Step 8),即可由当前最优个体给出子问题的一个近似解,因此保证 HBDEfRTVKP 能够求得 RTVKP 实例的一个近似解的充分条件是 $\Theta(n(N+\log n))<\mathbf{Min}\boldsymbol{T}$.

事实上,充分条件 $\Theta(n(N+\log n))<\mathbf{Min}\boldsymbol{T}$ 虽然能够保证 HBDEfRTVKP 求得 RTVKP 问题的近似解,但是这个近似解的近似程度却是难以确定的.一般地,进化算法的迭代进化次数越多,求得问题的近似解越好,甚至可能求得最优解.由于通常总有 $N<<n$,因此 HBDEfRTVKP 求解 RTVKP 的各子问题时,一次迭代进化的时间复杂度上界为 $O(n^2)$,其迭代进化一次的时间非常短,所以在求解各子问题时总能够迭代多次,从而求得较好的近似解或最优解.这一点将在第 4 节的实例计算中得到验证.

3.3 基于粒子群优化求解RTVKP问题

粒子群优化(PSO)[23]是由 Kennedy 和 Eberhart 提出的一种仿生进化算法,也是目前应用范围最广的进化算法之一[23,24,38−42].标准 PSO 仅适用于求解连续域上的最优化问题,为了能够求解离散域上的最优化问题,Kennedy 等人[42]首先提出了一种二进制粒子群优化算法(binary particle swarm optimization,简称 BPSO),但是 BPSO 在求解某些组合优化问题(如 SAT 问题)时效果并不理想.贺毅朝等人通过采用双重编码结构(其实质与 HBDE 的混合编码思想相同),在 BPSO 的基础上提出了一种改进的二进制粒子群优化算法:具有双重结构编码的二进制粒子群优化算法(DS_BPSO)[24],DS_BPSO 弥补了 BPSO 的种群多样性差和全局搜索能力相对低下的缺陷,在求解 0-1KP、随机 3-SAT 问题和 TVKP 时表现突出[24,30].

下面首先基于最大优化问题简述 DS_BPSO 的基本原理,然后给出它与 GOA 相结合求解 RTVKP 的进化算法的伪代码描述.

在 DS_BPSO 的第 t 代种群 $\boldsymbol{P}(t)$中,第 i 个粒子(即第 i 个个体)用有序三元组$((X_i(t),Y_i(t)),V_i(t))$表示,其中,$1\leqslant i\leqslant N$,N 为种群规模,$X_i(t)=(x_{i1}(t),x_{i2}(t),\ldots,x_{in}(t))$与 $V_i(t)=(v_{i1}(t),v_{i2}(t),\ldots,v_{in}(t))$分别为粒子的位置与速度,并且 $X_i(t),V_i(t)\in\prod_{j=1}^{n}[L_j,U_j]$,$n$ 为问题的维数,$L_j<U_j(1\leqslant j\leqslant n)$且为实数,$Y_i(t)=(y_{i1}(t),y_{i2}(t),\ldots,y_{in}(t))\in\{0,1\}^n$ 为 $X_i(t)$对应的二进制解向量,它表示问题的一个潜在解.

记$(Z_i(t),B_i(t))$为 $\boldsymbol{P}(t)$中第 i 个粒子的历史最好位置及其对应的二进制解向量,其中,$Z_i(t)=(z_{i1}(t),z_{i2}(t),\ldots,z_n(t))\in\prod_{j=1}^{n}[L_j,U_j]$, $B_i(t)=(b_{i1}(t),b_{i2}(t),\ldots,b_{in}(t))\in\{0,1\}^n$,则$(Z_i(t),B_i(t))\in\{(X_i(k),Y_i(k))|1\leqslant k\leqslant t\}$且$f(B_i(t))=\max\{f(Y_i(k))|1\leqslant k\leqslant t\}$.

记$(Z_g(t),B_g(t))$为 DS_BPSO 在前 t 代迭代中求得的全局最好位置及其对应的二进制解向量,即$(Z_g(t),B_g(t))\in\{(Z_i(t),B_i(t))|1\leqslant i\leqslant N\}$且 $f(B_g(t))=\max\{f(B_i(t))|1\leqslant i\leqslant N\}$,则由 $\boldsymbol{P}(t)$中第 i 个粒子$((X_i(t),Y_i(t)),V_i(t))$产生种群 $\boldsymbol{P}(t+1)$

© 中国科学院软件研究所 http://www.jos.org.cn

中第 i 个粒子$((X_i(t+1),Y_i(t+1)),V_i(t+1))$的计算公式由式(9)~式(11)给出.

$$v_{ij}(t+1)=v_{ij}(t)+c_1r_1(z_{ij}(t)-x_{ij}(t))+c_2r_2(z_{gj}(t)-x_{ij}(t)) \tag{9}$$

$$x_{ij}(t+1)=x_{ij}(t)+v_{ij}(t+1) \tag{10}$$

$$y_{ij}(t+1)=\begin{cases}1, & \text{if } sig(x_{ij}(t+1))\geqslant 0.5\\ 0, & \text{otherwise}\end{cases} \tag{11}$$

其中,$1\leqslant i\leqslant N$,$1\leqslant j\leqslant n$;r_1 和 r_2 分别是(0,1)中两个独立的随机实数,c_1 和 c_2 称为加速常数,一般均在 0~2 之间取值;$sig(x)=1/(1+e^{-x})$是模糊函数.

于是,利用第 4.2 节中的相关约定和表示,基于 DS_BPSO 与 GOA 相结合求解 RTVKP 问题的进化算法(记为 DPSOfRTVKP)的伪代码描述如下.

算法 6. DPSOfRTVKP.

输入:初始价值集 $V_0=\{v_{01},v_{02},\ldots,v_{0n}\}$;初始重量集 $W_0=\{w_{01},w_{02},\ldots,w_{0n}\}$;初始背包载重 C_0;参数 n,m,*Threshold*,A_v,B_v,A_w,B_w,A_c 和 B_c;种群规模 N,L_j 和 $U_j(1\leqslant j\leqslant n)$.

输出:RTVKP 的 m 个子问题的最优值和最优解.

1. $V\leftarrow V_0$; $W\leftarrow W_0$; $C\leftarrow C_0$; $k\leftarrow 0$;
2. $T\leftarrow GetPeriod(\ )$; $time\leftarrow SystemTime(\ )$;
3. **while** ($k<m$) **do**
4. Generate the initial population $\boldsymbol{P}(0)=\{((X_i(0),Y_i(0)),V_i(0))|1\leqslant i\leqslant N\}$ randomly;
5. $H[1\ldots n]\leftarrow QuickSort(\{v_i/w_i|v_i\in V,w_i\in W,1\leqslant i\leqslant n\})$;
6. **for** i=1 **to** N **do** $(Y_i(0),f(Y_i(0)))\leftarrow$***GOA***$(n,V,W,C,Y_i(0),H[1\ldots n])$;
7. **for** i=1 **to** N **do** $(Z_i(0),B_i(0))\leftarrow(X_i(0),Y_i(0))$; $t\leftarrow 0$;
8. Get $(Z_g(t),B_g(t))$ from $\{(Z_i(t),B_i(t))|1\leqslant i\leqslant N\}$ according to $f(B_i(t))$;
9. **while** ($SystemTime(\ )-time<T$) **do**
10. **for** i=1 to N **do**
11. **for** j=1 to n **do**
12. $v_{ij}(t+1)\leftarrow v_{ij}(t)+c_1r_1(z_{ij}(t)-x_{ij}(t))+c_2r_2(z_{gj}(t)-x_{ij}(t))$;
13. $x_{ij}(t+1)\leftarrow x_{ij}(t)+v_{ij}(t+1)$;
14. **if** $\text{sig}(x_{ij}(t+1))\geqslant 0.5$ **then** $y_{ij}(t+1)\leftarrow 1$ **else** $y_{ij}(t+1)\leftarrow 0$;
15. **endfor**
16. **if** ($SystemTime(\ )-time\geqslant T$) **then goto** 24;
17. $(Y_i(t+1),f(Y_i(t+1)))\leftarrow$***GOA***$(n,V,W,C,Y_i(t+1),H[1\ldots n])$;
18. **if** $f(Y_i(t+1))\geqslant f(B_i(t))$ **then** $(Z_i(t+1),B_i(t+1))\leftarrow(X_i(t+1),Y_i(t+1))$;
19. **else** $(Z_i(t+1),B_i(t+1))\leftarrow(Z_i(t),B_i(t))$;
20. **endfor**
21. $t\leftarrow t+1$;
22. Get $(Z_g(t),B_g(t))$ from $\{(Z_i(t),B_i(t))|1\leqslant i\leqslant N\}$ according to $f(B_i(t))$;
23. **endwhile**
24. **output** $(f(B_g(t)),B_g(t))$;
25. **if** ($k<m-1$) **then**
26. $V_0\leftarrow V$; $W_0\leftarrow W$; $C_0\leftarrow C$; $T\leftarrow GetPeriod(\ )$;
27. $(V,W,C)\leftarrow CheckVary(V_0,W_0,C_0)$; $time\leftarrow SystemTime(\ )$;
28. **end if**
29. $k\leftarrow k+1$;

© 中国科学院软件研究所 http://www.jos.org.cn

30. **end while**

31. **return** (“Success to solve”).

算法 6 也是一种随机近似算法,利用它求解 RTVKP 问题时,能否保证算法得到一个较为满意的近似解仍是算法的关键.注意到算法 6 中 Step 4,Step 6 和 Step 7 的时间复杂度均为 $\Theta(nN)$,Step 5 的时间复杂度为 $\Theta(n\log n)$,Step 8 的时间复杂度为 $\Theta(n+N)$,而对于 RTVKP 的每一个子问题,只要 DPSOfRTVKP 能够产生至少 1 代种群(即执行完 Step 4~Step 8)即可由当前最优个体给出问题的一个近似解,所以保证 DPSOfRTVKP 能够求得 RTVKP 的近似解的充分条件与 HBDEfRTVKP 的相同,也为 $\Theta(n(N+\log n))<\mathbf{Min}\boldsymbol{T}$.同样地,利用 DPSOfRTVKP 求解 RTVKP 的子问题时,一次迭代进化的时间复杂度的上界为 $O(n^2)$,因此其迭代进化一次的时间是非常短的,从而在求解 RTVKP 的各子问题时总能够较快地求得更好的近似解,甚至可能是最优解.

4 实例计算与比较

由于 HBDEfRTVKP,DPSOfRTVKP 和 EGAfRDKP[13]均为随机近似算法,比较其时间复杂度是没有意义的.因此,为了说明它们的高效性和实用性,下面利用文献[13]中的 fixRTVKP 实例 1~实例 3 以及 fitRTVKP 实例 4 和实例 5(其具体数据见 http://pan.baidu.com/s/1BDgn0)进行计算,并与 DPfRTVKP 和 GAAfRDKP[13]的计算结果进行比较.所有仿真计算使用微型计算机 DELL Intel(R) Pentium (R)4-CPU 1.69GHz,内存为 256M,操作系统为 MicroSoft Windows XP.利用 Visual C++ 6.0 进行编程,利用 MATLAB 7.1 绘制算法的平均收敛曲线.

3 种进化算法的种群规模 N 均为 50.在 HBDEfRTVKP 中,α=0.5,CR=0.3;在 DPSOfRTVKP 中,$c_1=c_2$=2.0;并且在 HBDEfRTVKP 和 DPSOfRTVKP 中均取 L_j=–5.0,U_j=5.0($1\leqslant j\leqslant n$).EGAfRDKP 的参数取值参见文献[13].

在表 1~表 5 中,给出了利用 HBDEfRTVKP,DPSOfRTVKP 和 EGAfRDKP 对实例 1~实例 5 的 20 次求解得到的最好值(即各子实例的最好值序列,用 Best 表示)、最差值(用 Worst 表示)、最好值在 20 次计算中出现的比例(用 Rate 表示)和 20 次计算结果的平均值(用 Mean 表示),并将以上结果与利用精确算法 DPfRTVKP 得到的最优值(用 Optimal 表示)和近似算法 GAAfRDKP 得到的近似值(用 Approx 表示)进行比较.

Table 1 Comparing results of HBDEfRTVKP, DPSOfRTVKP and EGAfRDKP for solving Instance 1

表 1 HBDEfRTVKP,DPSOfRTVKP 和 EGAfRDKP 求解实例 1 的结果比较

Algorithms	Results	$RTVKP_0$	$RTVKP_1$	$RTVKP_2$	$RTVKP_3$	$RTVKP_4$	$RTVKP_5$	$RTVKP_6$	$RTVKP_7$	$RTVKP_8$	$RTVKP_9$
DPfRTVKP	Optimal	3 103	3 882	3 458	4 638	4 495	4 319	3 554	3 975	3 891	3 871
GAAfRDKP	Approx	3 103	3 882	3 458	4 633	4 483	4 315	3 554	3 975	3 863	3 850
HBDEfRTVKP	Best	3 103	3 882	3 458	4 638	4 495	4 319	3 554	3 975	3 891	3 871
	Worst	3 103	3 882	3 458	4 638	4 495	4 319	3 554	3 975	3 891	3 871
	Mean	3 103	3 882	3 458	4 638	4 495	4 319	3 554	3 975	3 891	3 871
	Rate (%)	100	100	100	100	100	100	100	100	100	100
DPSOfRTVKP	Best	3 103	3 882	3 458	4 638	4 495	4 319	3 554	3 975	3 891	3 870
	Worst	3 103	3 882	3 458	4 638	4 495	4 319	3 554	3 975	3 891	3 870
	Mean	3 103	3 882	3 458	4 638	4 495	4 319	3 554	3 975	3 891	3 870
	Rate (%)	100	100	100	100	100	100	100	100	100	100
EGAfRDKP	Best	3 103	3 882	3 458	4 638	4 495	4 315	3 554	3 975	3 873	3 870
	Worst	3 103	3 882	3 458	4 638	4 495	4 315	3 554	3 975	3 873	3 870
	Mean	3 103	3 882	3 458	4 638	4 495	4 315	3 554	3 975	3 873	3 870
	Rate (%)	100	100	100	100	100	100	100	100	100	100

从表 1 中的计算结果可看出,HBDEfRTVKP 以 100%的概率求得实例 1 的最优值.DPSOfRTVKP 以 100%求得子实例 $RTVKP_0$~$RTVKP_8$ 的最优值,虽然不能求得最后一个子实例的最优值,但其求得的最好值与最优值之间相差仅为 1.EGAfRTVKP 除了子实例 $RTVKP_5$,$RTVKP_8$ 和 $RTVKP_9$ 以外,以 100%的概率求得其余子实例的最优值,即使对于子实例 $RTVKP_5$,$RTVKP_8$ 和 $RTVKP_9$,它求得的最好值与最优值之间的差值也是极小的.显然,HBDEfRTVKP 是 3 种进化算法中求解效果最佳的,DPSOfRTVKP 和 EGAfRTVKP 的求解效果次之,而且 3 种进化算法求得的最差值均明显优于 GGAfRDKP.

从表 2 中的计算结果可以看出,HBDEfRTVKP 以 100%的概率求得子实例 $RTVKP_0$,$RTVKP_3$,$RTVKP_5$~$RTVKP_8$ 的最优值,以至少 20%的概率求得子实例 $RTVKP_2$,$RTVKP_4$ 和 $RTVKP_9$ 的最优值,但对子实例 $RTVKP_1$,它所求得

© 中国科学院软件研究所 http://www.jos.org.cn

的最好值与最优值仅相差22.DPSOfRTVKP以100%的概率求得子实例$RTVKP_3$的最优值,以至少40%的概率求得子实例$RTVKP_0$~$RTVKP_2$,$RTVKP_4$,$RTVKP_6$~$RTVKP_8$的最优值,虽然不能求得子实例$RTVKP_5$和$RTVKP_9$的最优值,但其得到的最好值与最优值之间的差值仅为1和6.EGAfRTVKP以100%的概率求得子实例$RTVKP_2$和$RTVKP_6$的最优值,以至少20%的概率求得子实例$RTVKP_0$,$RTVKP_1$,$RTVKP_3$,$RTVKP_4$,$RTVKP_7$和$RTVKP_8$的最优值,虽然不能得到子实例$RTVKP_5$和$RTVKP_9$的最优值,但求得的最好值与最优值之间的差值仅为1和3.显然,HBDEfRTVKP求得实例2的最好值略优于DPSOfRTVKP和EGAfRTVKP,但是它们求得的平均值相差不大,且它们求得的最差值也均优于GGAfRDKP.

Table 2 Comparing results of HBDEfRTVKP, DPSOfRTVKP and EGAfRDKP for solving Instance 2

表2 HBDEfRTVKP,DPSOfRTVKP和EGAfRDKP求解实例2的结果比较

Algorithms	Results	$RTVKP_0$	$RTVKP_1$	$RTVKP_2$	$RTVKP_3$	$RTVKP_4$	$RTVKP_5$	$RTVKP_6$	$RTVKP_7$	$RTVKP_8$	$RTVKP_9$
DPfRTVKP	Optimal	5 375	6 419	5 880	6 366	6 917	5 607	6 193	6 449	5 449	6 704
GAAfRDKP	Approx	5 372	6 347	5 795	6 357	6 851	5 582	6 107	6 442	5 384	6 679
HBDEfRTVKP	Best	5 375	6 397	5 880	6 366	6 917	5 607	6 193	6 449	5 449	6 704
	Worst	5 375	6 397	5 852	6 366	6 915	5 607	6 193	6 449	5 449	6 698
	Mean	5 375	6 397	5 868.7	6 366	6 916.1	5 607	6 193	6 449	5 449	6 701.2
	Rate (%)	100	100	50	100	20	100	100	100	100	20
DPSOfRTVKP	Best	5 375	6 419	5 880	6 366	6 917	5 606	6 193	6 449	5 449	6 698
	Worst	5 373	6 390	5 878	6 366	6 916	5 603	6 164	6 446	5 447	6 697
	Mean	5 374.8	6 405.1	5 879.4	6 366	6 916.4	5 605.2	6 183.5	6 448.3	5 448.6	6 697.6
	Rate (%)	90	40	70	100	40	50	60	70	80	70
EGAfRDKP	Best	5 375	6 419	5 880	6 366	6 917	5 606	6 193	6 449	5 449	6 701
	Worst	5 374	6 413	5 880	6 364	6 915	5 602	6 193	6 446	5 448	6 693
	Mean	5 374.7	6 418	5 880	6 365	6 916.1	5 605.1	6 193	6 447.4	5 448.8	6 700.8
	Rate (%)	70	30	100	50	20	40	100	40	80	40

从表3中的计算结果可看出:HBDEfRTVKP以100%的概率求得子实例$RTVKP_0$,$RTVKP_1$,$RTVKP_3$~$RTVKP_6$,$RTVKP_8$和$RTVKP_9$的最优值,以至少25%的概率求得子实例$RTVKP_2$的最优值,虽然不能求得子实例$RTVKP_7$的最优值,但所得最好值与最优值仅相差1.DPSOfRTVKP以100%的概率求得子实例$RTVKP_1$,$RTVKP_3$,$RTVKP_5$和$RTVKP_8$的最优值,以至少20%的概率求得子实例$RTVKP_0$,$RTVKP_4$,$RTVKP_6$和$RTVKP_9$的最优值,虽然不能求得子实例$RTVKP_2$和$RTVKP_7$的最优值,但得到的最好值与最优值之间的差值也是很小的.EGAfRTVKP以100%的概率求得子实例$RTVKP_0$,$RTVKP_3$,$RTVKP_5$和$RTVKP_8$的最优值,以至少10%的概率求得子实例$RTVKP_1$,$RTVKP_2$,$RTVKP_4$,$RTVKP_6$和$RTVKP_9$的最优值,即使对于子实例$RTVKP_7$,求得的最好值与最优值之间的差值也是很小的.显然,HBDEfRTVKP,DPSOfRTVKP和EGAfRTVKP求解实例3的效果与求解实例2时是相同的.

Table 3 Comparing results of HBDEfRTVKP, DPSOfRTVKP and EGAfRDKP for solving Instance 3

表3 HBDEfRTVKP,DPSOfRTVKP和EGAfRDKP求解实例3的结果比较

Algorithms	Results	$RTVKP_0$	$RTVKP_1$	$RTVKP_2$	$RTVKP_3$	$RTVKP_4$	$RTVKP_5$	$RTVKP_6$	$RTVKP_7$	$RTVKP_8$	$RTVKP_9$
DPfRTVKP	Optimal	78 116	82 028	87 840	86 570	88 555	90 857	80 698	90 748	88 856	83 705
GAAfRDKP	Approx	77 873	81 761	87 524	86 225	88 249	90 549	80 494	90 447	88 542	83 433
HBDEfRTVKP	Best	78 116	82 028	87 840	86 570	88 555	90 857	80 698	90 747	88 856	83 705
	Worst	78 116	82 028	87 827	86 570	88 555	90 857	80 698	90 747	88 856	83 705
	Mean	78 116	82 028	87 830.3	86 570	88 555	90 857	80 698	90 747	88 856	83 705
	Rate (%)	100	100	25	100	100	100	100	100	100	100
DPSOfRTVKP	Best	78 116	82 028	87 827	86 570	88 555	90 857	80 698	90 736	88 856	83 705
	Worst	78 115	82 028	87 818	86 570	88 507	90 857	80 692	90 736	88 856	83 690
	Mean	78 115.4	82 028	87 825.1	86 570	88 545.9	90 857	80 696	90 736	88 856	83 701.8
	Rate (%)	40	100	70	100	50	100	20	100	100	30
EGAfRDKP	Best	78 116	82 028	87 840	86 570	88 555	90 857	80 698	90 736	88 856	83 705
	Worst	78 116	82 025	87 825	86 570	88 534	90 857	80 692	90 736	88 856	83 700
	Mean	78 116	82 027.4	87 837	86 570	88 550.6	90 857	80 694.6	90 736	88 856	83 704
	Rate (%)	100	80	80	100	20	100	10	100	100	40

从表4中的计算结果可以看出,HBDEfRTVKP以100%的概率求得子实例$RTVKP_0$,$RTVKP_3$,$RTVKP_5$,$RTVKP_6$和$RTVKP_8$的最优值,以至少50%的概率求得子实例$RTVKP_2$和$RTVKP_4$的最优值,虽然不能求得子实

© 中国科学院软件研究所 http://www.jos.org.cn

例 $RTVKP_1$,$RTVKP_7$ 和 $RTVKP_9$ 的最优值,但所得最好值与最优值之间最多仅相差 10. DPSOfRTVKP 以 100%的概率求得子实例 $RTVKP_0$,$RTVKP_3$,$RTVKP_5$ 和 $RRTVKP_8$ 的最优值,以 60%的概率求得子实例 $RTVKP_6$ 的最优值;虽然不能求得子实例 $RTVKP_1$,$RTVKP_2$,$RTVKP_4$,$RTVKP_7$ 和 $RTVKP_9$ 的最优值,但其得到的最好值与最优值之间的差值最多也仅相差 10.EGAfRTVKP 以 100%的概率求得子实例 $RTVKP_0$,$RTVKP_3$,$RTVKP_5$ 和 $RTVKP_6$ 的最优值,虽然不能得到子实例 $RTVKP_1$,$RTVKP_2$,$RTVKP_4$,$RTVKP_7$,$RTVKP_8$ 和 $RTVKP_9$ 的最优值,但求得的最好值与最优值之间的差值最多仅为 8.显然,3 种进化算法求解实例 4 的效果基本相同,且它们求得的最差值均优于 GAAfRDKP.

Table 4 Comparing results of HBDEfRTVKP, DPSOfRTVKP and EGAfRDKP for solving Instance 4

表 4 HBDEfRTVKP,DPSOfRTVKP 和 EGAfRDKP 求解实例 4 的结果比较

Algorithms	Results	$RTVKP_0$	$RTVKP_1$	$RTVKP_2$	$RTVKP_3$	$RTVKP_4$	$RTVKP_5$	$RTVKP_6$	$RTVKP_7$	$RTVKP_8$	$RTVKP_9$
DPfRTVKP	Optimal	202 145	199 700	194 211	190 275	187 040	200 771	198 494	195 035	204 849	200 156
GAAfRDKP	Approx	202 145	199 699	194 188	190 275	187 031	200 771	198 465	194 993	204 824	200 124
HBDEfRTVKP	Best	202 145	199 699	194 211	190 275	187 040	200 771	198 494	195 031	204 849	200 146
	Worst	202 145	199 699	194 188	190 275	187 032	200 771	198 494	195 031	204 849	200 146
	Mean	202 145	199 699	194 199.5	190 275	187 036.8	200 771	198 494	195 031	204 849	200 146
	Rate (%)	100	100	50	100	60	100	100	100	100	100
DPSOfRTVKP	Best	202 145	199 699	194 208	190 275	187 032	200 771	198 494	195 031	204 849	200 146
	Worst	202 145	199 699	194 202	190 275	187 032	200 771	198 473	195 021	204 849	200 138
	Mean	202 145	199 699	194 206.8	190 275	187 032	200 771	198 485.6	195 028	204 849	200 143
	Rate (%)	100	100	80	100	100	100	60	70	100	50
EGAfRDKP	Best	202 145	199 699	194 208	190 275	187 032	200 771	198 494	195 031	204 844	200 151
	Worst	202 145	199 699	194 208	190 275	187 032	200 771	198 494	195 031	204 841	200 151
	Mean	202 145	199 699	194 208	190 275	187 032	200 771	198 494	195 031	204 843.4	200 151
	Rate (%)	100	100	100	100	100	100	100	100	80	100

从表 5 中的计算结果可以看出,HBDEfRTVKP 以 100%的概率求得子实例 $RTVKP_0$~$RTVKP_8$ 的最优值,以 60%的概率求得子实例 $RTVKP_9$ 的最优值,求解效果非常好.DPSOfRTVKP 以 100%的概率求得子实例 $RTVKP_0$,$RTVKP_6$ 和 $RRTVKP_8$ 的最优值,以至少 20%的概率求得子实例 $RTVKP_1$~$RTVKP_5$ 和 $RTVKP_7$ 的最优值,虽然不能求得子实例 $RTVKP_9$ 的最优值,但其得到的最好值与最优值之间的差值仅为 7.EGAfRTVKP 以 100%的概率求得子实例 $RTVKP_2$~$RTVKP_4$,$RTVKP_6$~$RTVKP_8$ 的最优值,以 90%的概率求得子实例 $RTVKP_1$ 的最优值,虽然不能得到子实例 $RTVKP_0$,$RTVKP_5$ 和 $RTVKP_9$ 的最优值,但求得的最好值与最优值之间的差值最多仅为 7.显然,HBDEfRTVKP 的求解效果最佳,DPSOfRTVKP 和 EGAfRDKP 的求解效果稍差一点,且它们求得的最差值均优于 GAAfRDKP.

Table 5 Comparing results of HBDEfRTVKP, DPSOfRTVKP and EGAfRDKP for solving Instance 5

表 5 HBDEfRTVKP,DPSOfRTVKP 和 EGAfRDKP 求解实例 5 的结果比较

Algorithms	Results	$RTVKP_0$	$RTVKP_1$	$RTVKP_2$	$RTVKP_3$	$RTVKP_4$	$RTVKP_5$	$RTVKP_6$	$RTVKP_7$	$RTVKP_8$	$RTVKP_9$
DPfRTVKP	Optimal	292 894	290 866	289 291	295 630	291 582	295 111	280 923	285 841	283 044	292 923
GAAfRDKP	Approx	292 892	290 860	289 285	295 625	291 540	295 070	280 923	285 832	283 011	292 904
HBDEfRTVKP	Best	292 894	290 866	289 291	295 630	291 582	295 111	280 923	285 841	283 044	292 923
	Worst	292 894	290 866	289 291	295 630	291 582	295 111	280 923	285 841	283 044	292 922
	Mean	292 894	290 866	289 291	295 630	291 582	295 111	280 923	285 841	283 044	292 922.6
	Rate (%)	100	100	100	100	100	100	100	100	100	60
DPSOfRTVKP	Best	292 894	290 866	289 291	295 630	291 582	295 111	280 923	285 841	283 044	292 916
	Worst	292 894	290 851	289 286	295 625	291 565	295 101	280 923	285 838	283 044	292 912
	Mean	292 894	290 861.2	289 287.3	295 626.5	291 573.1	295 107	280 923	285 840.1	283 044	292 913.2
	Rate (%)	100	20	20	30	20	40	100	70	100	30
EGAfRDKP	Best	292 892	290 866	289 291	295 630	291 582	295 106	280 923	285 841	283 044	292 916
	Worst	292 892	290 863	289 291	295 630	291 582	295 102	280 923	285 841	283 044	292 912
	Mean	292 892	290 865.7	289 291	295 630	291 582	295 105.6	280 923	285 841	283 044	292 913.2
	Rate (%)	100	90	100	100	100	90	100	100	100	30

由表 1~表 5 中的计算结果可以看出,HBDEfRTVKP 的求解效果略优于 DPSOfRTVKP 和 EGAfRDKP,而且它们都明显优于 GAAfRDKP;3 种进化算法求得各实例的最好值和平均值均非常好,并且最好值与最优值之间

© 中国科学院软件研究所 http://www.jos.org.cn

相差极小;3 种进化算法的平均求解效果基本相同,并且与 RTVKP 实例的规模和数据大小无关.

为了进一步说明 HBDEfRTVKP,DPSOfRTVKP 和 EGAfRDKP 的高效性与实用性,图 1~图 5 中给出了它们求解各实例的平均收敛曲线.对每个子实例 $RTVKP_i(0\leqslant i\leqslant 9)$,在周期 T_{i+1} 内分别取 6 个采样点 $t=T_i,T_i+T_{i+1}/10,T_i+T_{i+1}/4,T_i+T_{i+1}/2,T_i+3T_{i+1}/4$ 和 T_i+T_{i+1} 以及它们对应的 $\sum_{k=1}^{20}f^k(B(t))\big/20$ 来绘制平均收敛曲线,其中 $T_0=0,f^k(B(t))$ 为算法在第 k 次计算子实例 $RTVKP_i$ 时在 t 时刻求得的全局最优值.

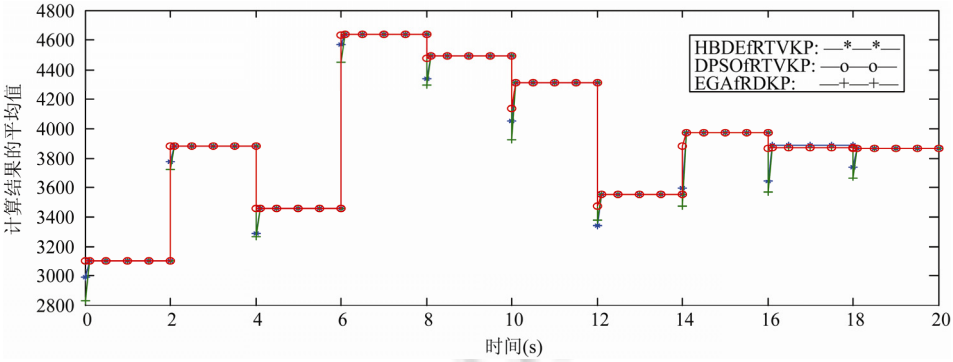

Fig.1 Average convergence curve of Instance 1 by HBDEfRTVKP, DPSOfRTVKP and EGAfRDKP

图 1 HBDEfRTVKP,DPSOfRTVKP 和 EGAfRDKP 求解实例 1 的平均收敛曲线

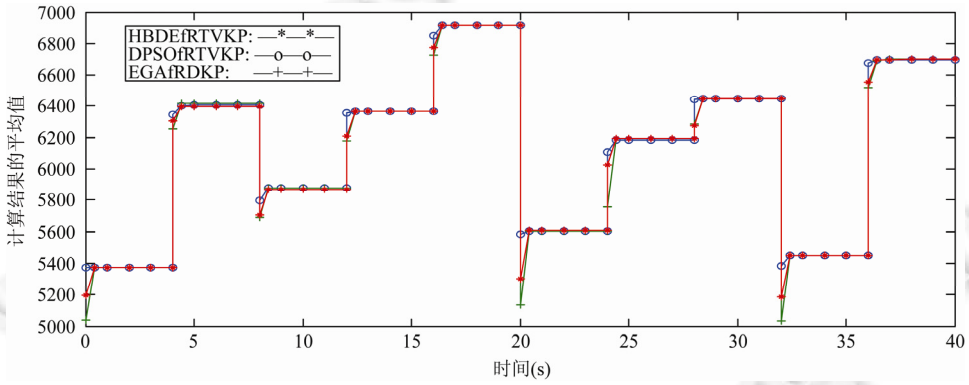

Fig.2 Average convergence curve of Instance 2 by HBDEfRTVKP, DPSOfRTVKP and EGAfRDKP

图 2 HBDEfRTVKP,DPSOfRTVKP 和 EGAfRDKP 求解实例 2 的平均收敛曲线

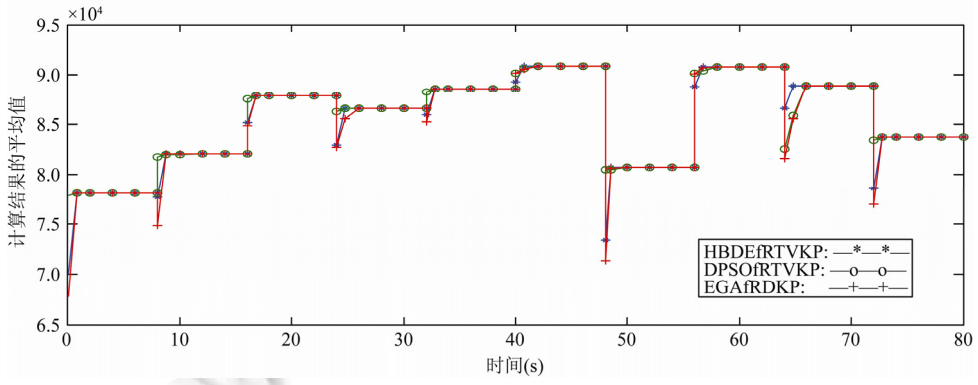

Fig.3 Average convergence curve of Instance 3 by HBDEfRTVKP, DPSOfRTVKP and EGAfRDKP

图 3 HBDEfRTVKP,DPSOfRTVKP 和 EGAfRDKP 求解实例 3 的平均收敛曲线

© 中国科学院软件研究所 http://www.jos.org.cn

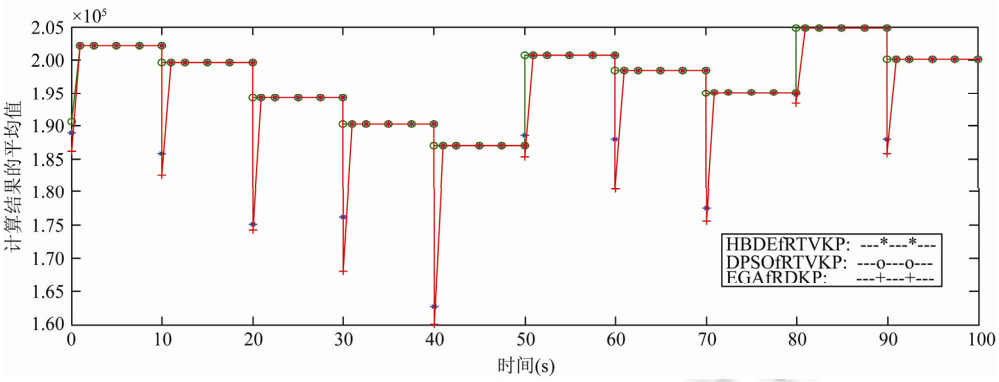

Fig.4 Average convergence curve of Instance 4 by HBDEfRTVKP, DPSOfRTVKP and EGAfRDKP

图 4 HBDEfRTVKP,DPSOfRTVKP 和 EGAfRDKP 求解实例 4 的平均收敛曲线

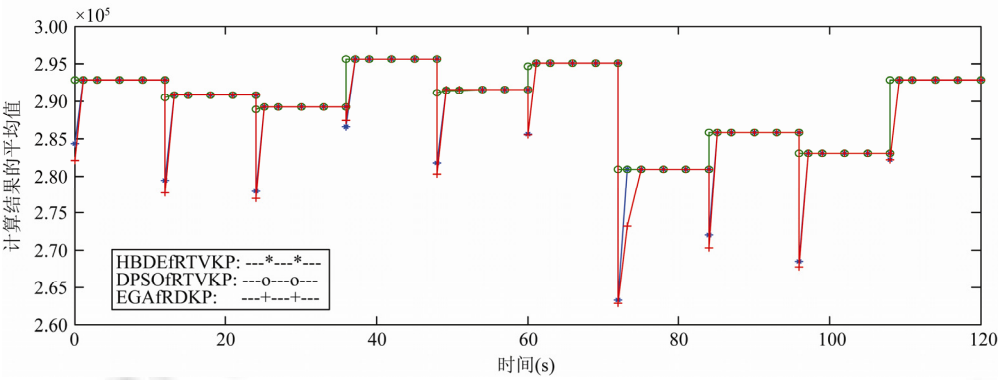

Fig.5 Average convergence curve of Instance 5 by HBDEfRTVKP, DPSOfRTVKP and EGAfRDKP

图 5 HBDEfRTVKP,DPSOfRTVKP 和 EGAfRDKP 求解实例 5 的平均收敛曲线

由图 1~图 5 中的平均收敛曲线不难看出,对于 RTVKP 的每一个子实例, HBDEfRTVKP,DPSOfRTVKP 和 EGAfRDKP 都能在不超过随机变化周期 1/10 的时间内求得一个很好的近似值,甚至是最优值,这表明 3 种算法只需要很少的迭代次数(当然也是很少的时间)就能求得 RTVKP 的较好近似解或最优解,因此它们对于具有较小随机变化周期的 RTVKP 问题都是非常有效和实用的.

通过对以上计算结果的比较和对平均收敛曲线的分析,我们可以得出以下结论.

结论 2. 对于实例 1~实例 5,HBDEfRTVKP,DPSOfRTVKP 和 EGAfRDKP 的各种求解结果均远远优于 GAAfRDKP.从算法求得最好值的能力来看,HBDEfRTVKP 比其他两种算法相对较强,而 DPSOfRTVKP 和 EGAfRDKP 的能力基本相当,但是从 3 种算法求得的平均值来看,它们的求解效果相差不大,这表明 3 种进化算法的平均求解能力基本上是相当的.

虽然理论上 DPfRTVKP 和 DPMfRDKP[13]都能求得 RTVKP 问题的最优解,但由于它们是伪多项式时间算法,当 RTVKP 实例的规模较大而且物品价值和重量均匀分布在一个大范围内时(例如,A_v=A_w=1,B_v=B_w=10^{12}),将导致所有物品价值之和与背包载重都非常大,使 DPfRTVKP 和 DPMfRDKP 求解 RTVKP 各子实例所耗费的时间巨大,从而导致求解失败.但是这种情况对于 HBDEfRTVKP,DPSOfRTVKP,EGAfRDKP 和 GAAfRDKP 是不存在的,因为可以通过放缩法对 RTVKP 实例中物品的价值、重量和背包载重统一进行适当的缩小,然后再利用它们进行求解,因此它们对于大规模且具有大数据的难 RTVKP 实例都是适用的,只是 GAAfRDKP 的求解效果比 3 种进化算法明显相差许多而已.

© 中国科学院软件研究所 http://www.jos.org.cn

事实上,对于进化算法而言,不存在最好,只有最适合.即不能简单地说某一进化算法最好,另一进化算法最差,只能是针对给定的具体问题,比较哪种进化算法最适宜求解,哪种进化算法不适宜求解.此外,在求解大规模NPC 问题时,如果找不到更有效的近似算法,那么利用进化算法求解往往是一种更明智的选择.

综上所述,容易给出以下的结论.

结论 3. 对于大规模、大数据且振荡频率较大的难 RTVKP 实例,精确算法和近似算法一般不适于求解,而算法 HBDEfRTVKP,DPSOfRTVKP 和 EGAfRDKP 不受这些因素的影响,它们不但求解速度快,而且还能获得非常好的近似解,甚至是最优解,因此它们是求解 RTVKP 问题的非常高效且实用的进化算法.

5 结束语

本文首先基于动态规划法提出了求解 RTVKP 问题的一种新精确算法 DPfRTVKP,通过与已有精确算法的时间复杂度比较指出,对于所有物品价值之和较小的一类 RTVKP 实例,DPfRTVKP 比 DPMfRDKP 的求解速度更快.然后,将差分演化和粒子群优化与贪心优化策略相结合给出了求解 RTVKP 问题的两种高效进化算法 HBDEfRTVKP 和 DPSOfRTVKP,并通过对 5 个 RTVKP 实例的计算与比较指出:HBDEfRTVKP,DPSOfRTVKP 和 EGAfRDKP 均是高效的,它们的平均求解能力基本相当,并且均比精确算法和近似算法更适于求解大规模、大数据且振荡频率较大的难 RTVKP 实例.

由于 RTVKP 问题是一种新的动态组合优化问题,其性质分析和算法设计研究还相对较少,关于其高效近似算法设计是其中研究的一个重点.为此,我们将进一步探讨利用其他进化算法(如和声搜索算法[12]、布谷鸟算法[43]等)求解 RTVKP 的有效方法.此外,虽然对实例的计算结果表明利用 3 种进化算法求解 RTVKP 比利用近似算法 GAAfRDKP 的求解结果更优,但是缺乏严谨的理论证明,因此,如何在理论上证明 3 种进化算法优于 GAAfRDKP,也是一个值得今后探讨的问题.

References:

[1] Du DZ, Ko KI. Theory of Computational Complexity. New York: Wiley-Interscience, 2000. 1−73.

[2] Gilmore PC, Gomory RE. The theory and computation of knapsack functions. Operations Research, 1966,14(6):1045−1074. [doi: 10.1287/opre.14.6.1045]

[3] Cord J. A method for allocating funds to investment projects when returns are subject to uncertainty. Management Science, 1964, 10(2):335−341. [doi: 10.1287/mnsc.10.2.335]

[4] Goldberg DE, Smith RE. Nonstationary function optimization using genetic algorithms with dominance and diploidy. In: Proc. of the Int'l Conf. on Genetic Algorithms. Hillsdale: L. Erlbaum Associates Inc., 1987. 59−68.

[5] Wang DW, Wang JW, Wang HF, Zhang RY, Guo Z. Intelligent Optimization Methods. Beijing: Higher Education Press, 2007. 282−302 (in Chinese).

[6] Hadad BS, Eick CF. Supporting polyploidy in genetic algorithms using dominance vectors. In: Proc. of the 6th Int'l Conf. on Evolutionary Computation. 1997. 223−234. [doi: 10.1007/BFb0014814]

[7] Yang S. Non-Stationary problem optimization using the primal-dual genetic algorithm. In: Proc. of the 2003 Congress on Evolutionary Computation. 2003. 2246−2253. [doi: 10.1109/CEC.2003.1299951]

[8] Lewis J, Hart E, Ritchie G. A comparison of dominance mechanisms and simple mutation on non-stationary problems. In: Proc. of the Parallel Problem Solving from Nature. 1998. 139−148. [doi: 10.1007/BFb0056857]

[9] Zhou CH, Xie AS. Dynamic niche-based self-organizing learning algorithm. Ruan Jian Xue Bao/Journal of Software, 2011,22(8): 1738−1748 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3830.htm [doi: 10.3724/SP.J.1001.2011.03830]

[10] He YC, Tian HY, Zhang XL, Wang ZW, Gao SG. Solving dynamic 0-1 knapsack problems based on dynamic programming algorithm. Computer Science, 2012,39(7):237−241 (in Chinese with English abstract).

[11] He YC, Zhang XL, Gao SG, Song C. An exact algorithm for solving random time-varying knapsack problem. Journal of Chinese Computer System, 2014,35(4):854−857 (in Chinese with English abstract).

[12] Li N, He YC, Tian HY. Application of hybrid coding harmony search algorithm in dynamic optimization. Computer Engineering, 2012,38(12):237−241 (in Chinese with English abstract).

© 中国科学院软件研究所 http://www.jos.org.cn

[13] He YC, Zhang XL, Li WB, Li X, Wu WL, Gao SG. Algorithms for randomized time-varying knapsack problems. Journal of Combinatorial Optimization, 2016,31(1):95−117. [doi: 10.1007/s10878-014-9717-1]

[14] Cormen TH, Leiserson CE, Rivest RL, Stein C. Introduction to Algorithms. 3rd ed., Cambridge: MIT Press, 2009.

[15] Levitin A. Introduction to the Design & Analysis of Algorithms. 3rd ed., Pearson Education, Inc., 2012.

[16] Kellerer H, Pferschy U, Pisinger D. Knapsack Problems. Berlin: Springer-Verlag, 2004.

[17] Xu ZB. Computational Intelligence—Simulated Evolutionary Computation. Beijing: Higher Education Press, 2005 (in Chinese).

[18] Li MQ, Kou JS, Lin D, Li SQ. The Foundational Theory and Application of Genetic Algorithms. Beijing: Science Press, 2003 (in Chinese).

[19] Storn R, Price K. Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization, 1997,11(4):341−359. [doi: 10.1023/A:1008202821328]

[20] He YC, Wang XZ, Liu KQ, Wang YQ. Convergent analysis and improvement of differential evolution. Ruan Jian Xue Bao/Journal of Software, 2010,21(5):875−885 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3486.htm [doi: 10.3724/SP.J.1001.2010.03486]

[21] He YC, Wang XZ, Kou YZ. A binary differential evolution algorithm with hybrid encoding. Journal of Computer Research and Development, 2007,44(9):1476−1484 (in Chinese with English abstract). [doi: 10.1360/crad20070905]

[22] Dorigo M, Stützle T. Ant Colony Optimization. Cambridge: MIT Press, 2004, 15−120.

[23] Poli R, Kennedy J, Blackwell T. Particle swarm optimization: An overview. Swarm Intelligence, 2007,1(1):33−57. [doi: 10.1007/s11721-007-0002-0]

[24] He YC, Wang YQ, Liu JQ. A new binary particle swarm optimization for solving discrete problems. Computer Applications and Software, 2007,24(1):157−159 (in Chinese with English abstract).

[25] Amiri B, Fathian M, Maroosi A. Application of shuffled frog-leaping algorithm on clustering. The Int'l Journal of Advanced Manufacturing Technology, 2009,45(1-2):199−209. [doi: 10.1007/s00170-009-1958-2]

[26] Geem ZW, Lee KS. A new meta-heuristic algorithm for continuous engineering optimization: Harmony search theory and practice. Computer Methods in Applied Mechanics and Engineering, 2005,194(36):3902−3933. [doi: 10.1016/j.cma.2004.09.007]

[27] Gottlieb J, Marchiori E, Rossi C. Evolutionary algorithms for the satisfiability problem. Evolutionary Computation, 2002,10(1):35−50. [doi: 10.1162/106365602317301763]

[28] Maity S, Roy A, Maiti M. A modified genetic algorithm for solving uncertain constrained solid travelling salesman problems. Computers & Industrial Engineerin, 2015,83(2):273−296. [doi: 10.1016/j.cie.2015.02.023]

[29] Shen XJ, Wang WW, Zheng BJ, Li YX. Modified particle swarm optimization for 0-1 knapsack problem. Computer Engineering, 2006,32(18):23−24, 38 (in Chinese with English abstract).

[30] Li N, He YC, Kou YZ. Solving dynamic knapsack problems based on double-structure coding PSO. Computer Engineering and Applications, 2012,48(7):39−42 (in Chinese with English abstract).

[31] Runarsson TP, Yao X. Stochastic ranking for constrained evolutionary optimization. IEEE Trans. on Evolutionary Computation, 2000,4(3):284−294. [doi: 10.1109/4235.873238]

[32] Coello CAC. Theoretial and numerical constraint-handling techniques used with evolutionary algorithm: A survey of the state of art. Computer Methods in Applied Mechanics and Engineering, 2002,191(11-12):1245−1287. [doi: 10.1016/S0045-7825(01)00323-1]

[33] Michalewicz Z, Schoenauer M. Evolutionary algorithms for constrained parameter optimization problems. Evolutionary Computation, 1996,4(1):1−32. [doi: 10.1162/evco.1996.4.1.1]

[34] Xue Y, Zhuang Y, Gu JJ, Chang XM, Wang Z. Strategy selecting problem of self-adaptive discrete differential evolution algorithm. Ruan Jian Xue Bao/Journal of Software, 2014,25(5):984−996 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4448.htm [doi: 10.13328/j.cnki.jos.004448]

[35] Zhang LB, Zhou CG, Ma M, Sun CT. A multi-objective differential evolution algorithm based on max-min distance density. Journal of Computer Research and Development, 2007,44(1):177−184 (in Chinese with English abstract). [doi: 10.1360/crad20070125]

[36] Das S, Suganthan PN. Differential evolution: A survey of the state-of-the-art. IEEE Trans. on Evolutionary Computation, 2011,15(1):4−31. [doi: 10.1109/TEVC.2010.2059031]

[37] Wang Y, Cai ZX, Zhang QF. Differential evolution with composite trial vector generation strategies and control parameters. IEEE Trans. on Evolutionary Computation, 2011,15(1):55−66.

[38] Mao CY, Yu XX, Xue YZ. Algorithm design and empirical analysis for particle swarm optimization-based test data generation. Journal of Computer Research and Development, 2014,51(4):824−837 (in Chinese with English abstract).

© 中国科学院软件研究所 http://www.jos.org.cn

[39] Li XD, Yao X. Cooperatively coevolving particle swarms for large scale optimization. IEEE Trans. on Evolutionary Computation, 2012,16(2):210−224. [doi: 10.1109/TEVC.2011.2112662]

[40] Li XD. Niching without niching parameters: Particle swarm optimization using a ring topology. IEEE Trans. on Evolutionary Computation, 2010,14(1):150−169. [doi: 10.1109/TEVC.2009.2026270]

[41] Chen X, Gu Q, Wang ZY, Chen DX. Framework of particle swarm optimization based pairwise testing. Ruan Jian Xue Bao/Journal of Software, 2011,22(12):2879−2893 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3973.htm [doi: 10.3724/SP.J.1001.2011.03973]

[42] Kennedy J, Eberhart RC. A discrete binary version particle swarm optimization. In: Proc. of the 1997 Conf. on System, Man, and Cybernetices. Piscataway: IEEE Service Center, 1997. 4104−4109. [doi: 10.1109/ICSMC.1997.637339]

[43] Yang XS, Deb S. Engineering optimization by cuckoo search. Int'l Journal of Mathematical Modelling and Numerical Optimization, 2010,1(4):330−343. [doi: 10.1504/IJMMNO.2010.035430]

附中文参考文献:

[5] 汪定伟,王俊伟,王洪峰,张瑞友,郭哲.智能优化方法.北京:高等教育出版社,2007.

[9] 周传华,谢安世.一种基于动态小生境的自组织学习算法.软件学报,2011,22(8):1738−1748. http://www.jos.org.cn/1000-9825/3830.htm [doi: 10.3724/SP.J.1001.2011.03830]

[10] 贺毅朝,田海燕,张新禄,王志威,高锁刚.基于动态规划法求解动态 0-1 背包问题.计算机科学,2012,39(7):237−241.

[11] 贺毅朝,张新禄,高锁刚,宋超.求解随机时变背包问题的确定性算法.小型微型计算机系统,2014,35(4):854−857.

[12] 李宁,贺毅朝,田海燕.混合编码和声搜索算法在动态优化中的应用.计算机工程,2012,38(12):237−241.

[17] 徐宗本.计算智能——模拟进化计算.北京:高等教育出版社,2005.

[18] 李敏强,寇纪松,林丹,李书全.遗传算法的基本理论与应用.北京:科学出版社,2003.

[20] 贺毅朝,王熙照,刘坤起,王彦祺.差分演化的收敛性分析与算法改进.软件学报,2010,21(5):875−885. http://www.jos.org.cn/1000-9825/3486.htm [doi: 10.3724/SP.J.1001.2010.03486]

[21] 贺毅朝,王熙照,寇应展.一种具有混合编码的二进制差分演化算法.计算机研究与发展,2007,44(9):1476−1484. [doi: 10.1360/crad20070905]

[24] 贺毅朝,王彦祺,刘建芹.一种适于求解离散问题的二进制粒子群优化算法.计算机应用与软件,2007,24(1):157−159.

[29] 沈显君,王伟武,郑波尽,李元香.基于改进微粒群优化算法的 0-1 背包问题求解.计算机工程,2006,32(18):23−24,38.

[30] 李宁,贺毅朝,寇应展.利用双重结构编码 PSO 求解动态背包问题.计算机工程与应用,2012,48(7):39−42.

[34] 薛羽,庄毅,顾晶晶,常相茂,王洲.自适应离散差分进化算法策略的选择.软件学报,2014,25(5):984−996. http://www.jos.org.cn/1000-9825/4448.htm [doi: 10.13328/j.cnki.jos.004448]

[35] 张利彪,周春光,马铭,孙彩堂.基于极大极小距离密度的多目标微分进化算法.计算机研究与发展,2007,44(1):177−184. [doi: 10.1360/crad20070125]

[38] 毛澄映,喻新欣,薛云志.基于粒子群优化的测试数据生成及其实证分析.计算机研究与发展,2014,51(4):824−837.

[41] 陈翔,顾庆,王子元,陈道蓄.一种基于粒子群优化的成对组合测试算法框架.软件学报,2011,22(12):2879−2893. http://www.jos.org.cn/1000-9825/3973.htm [doi: 10.3724/SP.J.1001.2011.03973]

贺毅朝(1969－),男,河北晋州人,教授,CCF 高级会员,主要研究领域为进化算法及其应用,算法设计与分析,群测试理论.

李文斌(1974－),男,博士,教授,CCF 高级会员,主要研究领域为机器学习,演化计算,数据挖掘.

王熙照(1963－),男,博士,教授,博士生导师,主要研究领域为机器学习,进化计算,大数据分析.

赵书良(1967－),男,博士,教授,博士生导师,CCF 专业会员,主要研究领域为智能信息处理.

© 中国科学院软件研究所 http://www.jos.org.cn