求解二次分配问题的离散粒子群优化算法

钟一文 [1] 蔡荣英 [1]

摘 要 提出了一种求解二次分配问题的离散粒子群优化算法. 根据二次分配问题及离散量的特点, 重新定义了粒子的位置、速度等量及其运算规则, 为抑制早熟停滞现象, 为粒子和粒子群分别定义了个体多样性和平均多样性. 算法中定义了排斥算子来保持粒子群的多样性, 使用局部搜索算子来提高算法的局部求精能力, 使算法在空间勘探和局部求精间取得了较好的平衡. 在 QAPLIB 的实例上的仿真结果表明, 离散粒子群优化算法具有良好的性能.

关键词 离散粒子群优化, 二次分配问题, 排斥算子, 局部搜索算子
中图分类号 TP301

Discrete Particle Swarm Optimization Algorithm for QAP

ZHONG Yi-Wen[1] CAI Rong-Ying[1]

Abstract A discrete particle swarm optimization algorithm is presented to tackle the quadratic assignment problem (QAP). Based on the characteristics of QAP and discrete variable, this paper redefines particles′ position, velocity, and their operation rules. In order to restrain premature stagnation, individual-diversity of particle and average-diversity of particle swarm are defined. A repulsion operator is designed to keep the diversity of particle swarm, and an efficient local search operator is used to improve the algorithm′s intensification ability. Using those operators, the proposed algorithm can get good balance between exploration and exploitation. Experiments were performed on QAP instances from QAPLIB. The simulation results show that it can produce good results.

Key words Discrete particle swarm optimization, quadratic assignment problem, repulsion operator, local search operator

1 引言

粒子群优化 (Particle swarm optimization, PSO) 算法是一种典型的计算智能方法, 它最早由 Kennedy 和 Eberhart[1, 2] 提出, 其基本原理源于对鸟群捕食行为的仿真. 与蚁群优化 (Ant colony optimization, ACO) 算法类似, PSO 算法是一种基于群智能方法的优化技术, 同时还与遗传算法类似, 是一种基于进化的优化工具.

目前对 PSO 算法的研究主要集中在连续型的 PSO 算法, 即描述粒子状态及其运动规律的量都是连续的, 而对离散粒子群优化 (Discrete particle swarm optimization, DPSO) 算法的研究甚少. PSO 算法在函数优化等领域取得的巨大成功, 使许多学者试图用它去解决组合优化问题, 比如, 为解决工程实际中的组合优化问题, 文献 [3] 提出 PSO 算法的离散二进制版, 把位置表示为离散型的 0 和 1, 但速度还是连续型的量; 文献 [4] 使用 PSO 算法去解决单一机器调度问题, 它采用随机键表示法表示粒子的位置, 再根据每一维的值对作业进行排序, 就可以把位置映射为合法的调度; 文献 [5, 6] 也用相同的方式把 PSO 算法应用到置换 flow-shop 问题及单一机器带权总拖期调度问题的求解中; 文献 [7] 用 PSO 算法去解带时间窗车辆路径问题, 位置和速度都表示为整数, 但运算还是采用连续量的运算法则, 而对运算的结果向上取整, 超过范围时按边界取值. 遗憾的是, 这些方法并没有充分考虑到离散型组合优化的特点, 不可避免地存在表示冗余大、搜索效率低等缺点[4].

本文以二次分配问题 (Quadratic assignment problem, QAP) 为例, 提出了一种 DPSO 算法, 根据 QAP 及离散量运算的特点, 对粒子的位置、速度等量及其运算规则进行了重新定义, 在典型 QAP 实例上的仿真表明, DPSO 算法能够产生良好的结果.

2 二次分配问题

描述 QAP 为: 给定 n 个设施和 n 个地点, 定义两个 $n\times n$ 矩阵, 即 $A=(a_{ij})$ 和 $B=(b_{ij})$, 其中 a_{ij} 是设施 i 和 j 之间的流量, b_{ij} 是地点 i 和 j 之间的距离. 要求给每个设施分配一个地点, 令 $P:\{1,2,\ldots,n\}\rightarrow\{1,2,\ldots,n\}$ 是一个分配方案, 定义其代价为

$$c(P)=\sum_{i=1}^{n}\sum_{j=1}^{n}(a_{ij}b_{p_ip_j}) \tag{1}$$

式中 p_i 和 p_j 分别表示设施 i 和 j 被分配的地点. 优化的目标是找到一个排列 $P=(p_1,p_2,\ldots,p_n)$, $P\in\prod(n)$, 使其代价最小, 即 Minimize($c(P)$).

目前, QAP 已经得到了深入的研究, 由于它是一个 NP-困难的问题, 文献中存在许多启发式方法去求取近优解, 比如, 快速蚁群算法 (Fast ant colony algorithm, FANT)[8, 9] 和近似骨架导向快速蚁群算法 (Approximate-backbone guided fast ant colony algorithm, ABFANT)[10], 是目前基于 ACO 算法的求解 QAP 的优秀算法.

3 粒子群优化算法

根据粒子对粒子群中的其它粒子的感知程度, 可以把 PSO 算法分为全局和局部 PSO 算法, 全局 PSO 算法中的每个粒子知道粒子群中所有粒子中的历史最好解, 而局部 PSO 算法中的粒子只知道某种近邻结构中的最好解. PSO 算法在每一次迭代中, 粒子通过跟踪两个 "极值" 来更新自己: 第一个就是粒子本身所找到的最好解, 叫做个体极值点 (用 *pbest* 表示其位置), 全局版 PSO 算法中的另一个极值点是整个种群目前找到的最好解, 称为全局极值点 (用 *gbest* 表示其位置), 而局部版 PSO 算法不用整个种群而是用其中一部分作为粒子的邻居, 所有邻居中的最好解就是局部极值点 (用 *lbest* 表示其位置). 粒子的信息可以用 N 维向量表示, 把位置表示为 $\boldsymbol{x}=(x_1,\ldots,x_n)$, 而把速度表示为 $\boldsymbol{v}=(v_1,\ldots,v_n)$, 其他向量类似. 式 (2) 和 (3) 是粒子速度和位置的更新方程 (对局部 PSO 算法, 用 *lbest* 替换 (2) 中的 *gbest*):

$$v_j^{t+1}=wv_j^t+c_1r_1(pbest_j^t-x_j^t)+c_2r_2(gbest_j^t-x_j^t) \tag{2}$$

收稿日期 2006-1-18 收修改稿日期 2006-5-13
Received January 18, 2006; in revised form May 13, 2006
福建省自然科学基金 (A0540006), 福建省青年人才科技创新基金(2006F3013) 资助
Supported by the Natural Science Foundation of Fujian Province (A0540006), and Fujian Provincial Innovation Foundation for Young Science and Technology Talents (2006F3013)
1. 福建农林大学计算机与信息学院 福州 350002
1. College of Computer and Information, Fujian Agriculture and Forestry University, Fuzhou 350002
DOI: 10.1360/aas-007-0871

$$x_j^{t+1} = x_j^t + v_j^{t+1} \tag{3}$$

式中 w 是惯性系数, 其主要作用是产生扰动, 以防止算法的早熟收敛; c_1 和 c_2 是加速系数, 分别调节向个体极值点和全局极值点方向飞行的最大步长; r_1 和 r_2 是 [0,1] 区间的随机数.

Clerc[11] 首次说明了建立 DPSO 算法的关键是为问题域定义与 DPSO 算法相关的数学对象及其运算规则. 作为一个具体例子, Clerc 针对旅行商问题问题实现了一个具体的 DPSO 算法, 它把位置定义为 N 个城市的排列, 而定义速度为交换的列表, 每个交换表示交换排列中的两个城市, 在此基础上, 对其它量及运算法则进行了定义, 由于缺少对离散量与连续量运算规律不同的考虑, 仿真结果表明它与其他算法相比仍有较大的差距[11].

4 求解 QAP 的 DPSO 算法

本文提出的求解 QAP 的 DPSO 算法, 采用与文献 [11] 相同的方式去表示粒子的位置 (N 个自然数的排列), 但速度的表示以及速度相关的运算法则完全不同.

4.1 粒子的位置

粒子的位置 $\boldsymbol{x}$ 是一个 N 维向量 (它是 N 个自然数的一个排列, 直接表示一种分配方案), 在 $\boldsymbol{x}$ 中, 维表示设施, 每一维的分量表示此设施的放置地点, 可以把 $\boldsymbol{x}$ 表示为

$$\boldsymbol{x} = (x_1, \ldots, x_i, \ldots, x_N), \quad 1 \le i \le N, \ 1 \le x_i \le N \tag{4}$$

4.2 粒子的速度

速度 $\boldsymbol{v}$ 的作用是改变粒子的位置, 与 $\boldsymbol{x}$ 类似, 速度 $\boldsymbol{v}$ 是一个 N 维向量, 表示为

$$\boldsymbol{v} = (v_1, \ldots, v_i, \ldots, v_N), \quad 1 \le i \le N, \ 0 \le v_i \le N \tag{5}$$

式中 N 是设施数. 同样, 速度 $\boldsymbol{v}$ 的维是设施, 但是每一维上的分速度 v_i 有二种含义, 如果 v_i 等于 0, 表示空操作, 即如果用该分速度作用某一个位置 $\boldsymbol{x}$, 它将不影响位置上相应维的数据 x_i, 而如果 v_i 不等于 0, 表示把此位置的相应维上的数据修改为 v_i, 它实际上是一种交换操作, 即交换 $\boldsymbol{x}$ 中的地点 x_i 和 v_i, 这样就保证了 $\boldsymbol{x}$ 在任意 $\boldsymbol{v}$ 的作用下依然是 N 个地点的一个排列, 即保证了解的可行性.

4.3 位置与速度的加法运算

位置与速度的加法运算实现了粒子位置的移动, 使粒子进入了一个新的位置, 表示为 $\boldsymbol{x} = \boldsymbol{x} + \boldsymbol{v}$. 新位置的每一维的数据依次由式 (6) 的操作确定:

$$\begin{cases} \emptyset & \text{若 } v_i = 0 \\ \text{swap}(x_i, v_i) & \text{否则} \end{cases} \tag{6}$$

式中 $\text{swap}(x_i, v_i)$ 表示交换 $\boldsymbol{x}$ 中的地点 x_i 和 v_i.

4.4 位置的减法

两个位置 $\boldsymbol{x}_2$ 和 $\boldsymbol{x}_1$ 相减的结果是一个速度 $\boldsymbol{v}$, 表示为 $\boldsymbol{v} = \boldsymbol{x}_2 - \boldsymbol{x}_1$. 它表示如果把 $\boldsymbol{v}$ 作用在 $\boldsymbol{x}_1$ 上将得到位置 $\boldsymbol{x}_2$, 它可以通过下述过程来计算: 比较 $\boldsymbol{x}_1$ 和 $\boldsymbol{x}_2$ 在每一维 i 上的分量, 即 $x_{1,i}$ 和 $x_{2,i}$, 如果相同, 则使相应维的分速度 v_i 等于 0, 如果不同, 则使 v_i 等于 $x_{2,i}$, 即 $\boldsymbol{v}$ 中的任意维的分速度可表示为

$$v_i = \begin{cases} 0 & \text{若 } x_{1,i} = x_{2,i} \\ x_{2,i} & \text{否则} \end{cases} \tag{7}$$

4.5 速度的数乘

速度的数乘具有概率意义, 它可以表示为 $\boldsymbol{v}_2 = c \cdot \boldsymbol{v}_1$. 式中 c 是一个常数, $c \in [0, 1]$, 它具有概率的意义, 在实际计算 $\boldsymbol{v}_2$ 时, 对 $\boldsymbol{v}_1$ 中的每一维的分速度 $v_{1,i}$, 生成一个 0 到 1 之间的随机数 $rand$, 如果 $rand$ 小于 c, 则使 $\boldsymbol{v}_2$ 在相应维的分速度 $v_{2,i}$ 等于 $v_{1,i}$, 否则, $v_{2,i}$ 等于 0, 即 $\boldsymbol{v}_2$ 中的任意一个元素可表示为

$$v_{2,i} = \begin{cases} v_{1,i} & \text{若 } rand < c \\ 0 & \text{否则} \end{cases} \tag{8}$$

4.6 速度的加法

两个速度相加得到的新速度表示为 $\boldsymbol{v} = \boldsymbol{v}_1 + \boldsymbol{v}_2$, 其中每一个速度分量 v_i 定义为

$$v_i = \begin{cases} v_{1,i} & \text{若 } (v_{1,i} \ne 0 \wedge v_{2,i} = 0) \vee \\ & \quad (v_{1,i} \ne 0 \wedge v_{2,i} \ne 0 \wedge rand < 0.5) \\ v_{2,i} & \text{否则} \end{cases} \tag{9}$$

式中 $rand$ 是一个 0 到 1 之间的随机数. 式 (9) 表示在任一维 i, 如果 $\boldsymbol{v}_1$ 和 $\boldsymbol{v}_2$ 中有且仅有一个的分速度为 0, 则取不为 0 的值, 如果两者都不为 0, 则以等概率从中随机取一个. 在加法运算中引入的随机数有助于维持粒子群的多样性, 当然, 这也使加法运算不再满足交换律.

4.7 粒子的运动方程

由于离散量运算的特殊性, 对粒子的运动方程作了修改, 取消了原有的惯性项, 即式 (2) 中右边的第一项, 因为速度的定义使位置的运动是一步到位的, 所以它意义不大, 粒子的运动方程具体描述如下

$$\boldsymbol{v} = c_1(\boldsymbol{x}_{pbest} - \boldsymbol{x}) + c_2(\boldsymbol{x}_{gbest} - \boldsymbol{x}) \tag{10}$$

$$\boldsymbol{x} = \boldsymbol{x} + \boldsymbol{v} \tag{11}$$

由于惯性的主要作用是产生扰动, 维持粒子群的多样性, 没有它的作用, DPSO 算法将迅速陷入早熟, 为解决这问题, 本文后面定义了排斥算子来维持粒子群的多样性, 使算法在迭代的后期能依然保持进化能力.

5 优化算子

PSO 算法应用于优化问题求解, 可以看作是一种随机化搜索过程, 在该搜索过程中, PSO 不仅要尽量保证探索解空间的全局性, 而且应当充分利用已经获得的解空间信息逼近当前局部最优解. 尽管 PSO 算法具有通用性的一面, 但却忽视了问题特征信息的辅助作用, 使得算法的局部求精能力不强; PSO 算法的另一个引起广泛关注的问题是早熟收敛, 研究表明, 早熟收敛总是与粒子群中粒子趋同、粒子群多样性的迅速下降有密切关系. 为了使 DPSO 算法在空间探索和局部求精间取得更好的平衡, 本文定义了排斥算子来保持粒子群的多样性, 使用局部搜索算子来提高局部求精能力.

5.1 基本概念

1) 粒子位置相似性 $s_{i,j}$: 是指任意两个位置 $\boldsymbol{x}_i$ 和 $\boldsymbol{x}_j$ 的相似程度, 定义为

$$s_{i,j} = \frac{1}{N} \sum_{k=1}^{N} iif(x_{i,k} = x_{j,k}, 1, 0) \tag{12}$$

式中 N 是位置的维数, 函数 iff 表示如果第一个参数为真, 则函数值等于第二个参数, 即 1, 否则, 函数值等于第三个参数, 即 0. 所以, $s_{i,j}$ 实际描述了在位置 $\boldsymbol{x}_i$ 和 $\boldsymbol{x}_j$ 中数据相同的维所占的比例, 如果 $s_{i,j}$ 等于 1, 则所有维的数据都相同, 如果等于 0, 则所有维的数据都不同, 也就是说 $s_{i,j} \in [0,1]$.

2) 粒子个体多样性 d_i: 是指粒子 $\boldsymbol{x}_i$ 和其历史最佳及全局最佳间的相异程度, 定义为

$$d_i = 1 - \frac{1}{3}(s_{i,pbest} + s_{i,gbest} + s_{pbest,gbest}) \tag{13}$$

如果 $d_i = 0$, 则三者完全相同, 如果 $d_i = 1$, 则三者两两完全不同, 也就是说 $d_i \in [0,1]$.

3) 粒子群平均多样性 $\overline{d}$: 是指个体多样性的平均值, 表示为

$$\overline{d} = \frac{1}{size}\sum_{i=1}^{size} d_i \tag{14}$$

式中 $size$ 是粒子群的大小. 在后面的仿真实验中将使用 $\overline{d}$ 来描述粒子群的多样性.

5.2　排斥算子

当粒子的个体多样性下降到一定程度后, 个体的进化能力就受到了很大的限制, 只有在其它粒子找到新的全局最佳位置后, 粒子才恢复进化能力, 所以, 在个体多样性下降到一定程度后, 必须有相应的算子去增加其多样性, 以保持个体的进化能力, 为此, 定义一个排斥算子, 对位置中的每一维数据, 如果它与历史最佳值 (或全局最佳值) 相同, 则以一定的概率为其随机生成一个新的速度, 再按式 (6) 把此速度作用到位置上.

5.3　局部搜索算子

基于种群的智能优化方法通常具有较好的空间勘探能力, 而局部求精能力比基于局部搜索的启发式算法差, 这两种方法的混合往往能产生更好的结果, 比如, FANT 和 ABFANT 算法中都采用了局部搜索算子来提高算法的性能, DPSO 算法也采用了相同的局部搜索算子, 局部搜索算子的算法描述详见文献 [9].

6　仿真实验及结果

为了检验 DPSO 算法的性能, 与文献 [10] 相同, 从通用 QAPLIB[12] 中选用典型的 Benchmark 实例进行了仿真, 分别对排斥算子的作用和算法的总体性能进行了仿真比较.

6.1　排斥算子的作用和算法的收敛过程

为了观察排斥算子对粒子群多样性的作用效果, 比较了不带排斥算子与带排斥算子的情况下粒子群平均多样性的变化情况, 仿真中 c_1 和 c_2 的值均设为 0.1, 在个体多样性低于 0.2 时启用排斥算子, 图 1 是在 Tai30b 实例上的仿真结果, 从图中可以看出, 如果不使用排斥算子, 则粒子群很快趋同, 失去了进化能力, 排斥算子能有效地维持粒子群的多样性, 具体数值可通过启用时机和排斥概率来设定, 由于有效地避免了早熟收敛, 仿真发现, 使用排斥算子后的结果好于不使用排斥算子的结果.

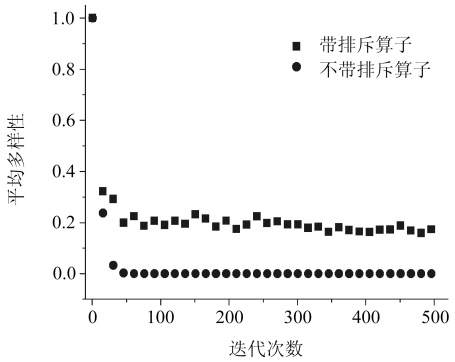

图 1　在 Tai30b 实例上粒子群 $\overline{d}$ 的变化曲线

Fig. 1　The changing process of $\overline{d}$ of particle swarm on Tai30b instance

图 2 是 DPSO 算法在 Tai30b 实例上所获取的最佳值和平均值随迭代次数变化的情况, 从图中可以看出, 算法的收敛性能比较理想, 从平均值的变化曲线可以看出, 即使在迭代的后期, 也不会出现所有粒子趋同的现象, 使粒子群在迭代后期依然存在继续进化的能力, 这也进一步说明了排斥算子对维持 DPSO 算法进化能力的作用.

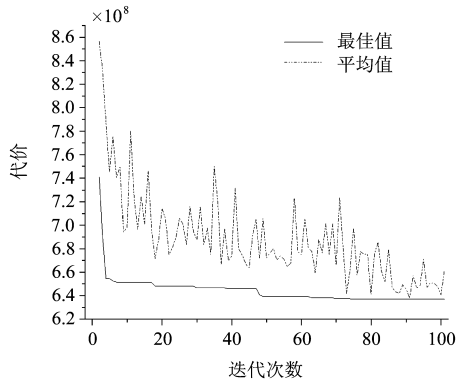

图 2　在 Tai30b 实例上最佳值和平均值的收敛过程

Fig. 2　The convergence process of best value and average value on Tai30b instance

6.2　算法的总体效果

对 DPSO 算法的性能进行了对比实验, 在实验中, 使用全局版 PSO 算法, c_1 和 c_2 均设置为 0.1, 由于采用了排斥算子和局部搜索算子, 粒子群可以比较小, 仿真中设为 10, 当个体多样性小于 0.1 时使用排斥算子. 从文献中选用了 FANT 和 ABFANT 算法进行比较, 由于 FANT 和 ABFANT 算法的蚂蚁数均为 1, 而 DPSO 算法的粒子数为 10, 为了进行公平的比较, 三种算法调用相同次数的局部搜索算子.

在文献 [10] 所使用的 QAP 问题上对算法性能进行了测试, 表 1(见下页) 是测试结果, 其中 FANT 和 ABFANT 算法的数据均来源于文献 [10], 表中迭代次数是指 FANT 和 ABFANT 算法的迭代次数, 由于 FANT 和 ABFANT 算法每次迭代要二次调用局部搜索算子, 而 DPSO 算法有 10 个粒子, 每个粒子在每代调用 1 次局部搜索算子, 所以, DPSO 算法的迭代次数是 FANT 算法的迭代次数的五分之一, 比

表 1 仿真结果与 FANT 和 ABFANT 算法的比较
Table 1 Simulation results comparing with FANT and ABFANT algorithm

QAP 实例	已知最佳值	FANT	ABFANT	DPSO	迭代次数
Tai30b	637117113	**637141763**	637550702	637166983	350
Tai40b	637250948	638672951	638030216	**637536295**	350
Tai50b	458821517	460061936	459994458	**459016693**	850
Chr22b	6194	6310.9	**6296.9**	6306.2	750
Chr25a	3796	4220.7	4160.5	**4027.4**	350
Kra30a	88900	90309.0	90246.3	**89763.0**	300
Kra30b	91420	91782	91723.6	**91681.0**	350
Wil50	48816	48916.2	48914.3	**48887.0**	400
Esc32a	130	135.6	**134.7**	136.0	350
Ste36a	9526	9612.7	**9603.1**	9634.0	350
Lipa40a	31538	31831.4	31827.8	**31798.1**	400
Sko42	15812	15857.2	15867.0	**15853.2**	650
Sko64	48498	48576.4	48564.3	**48536.6**	2500
Sko72	66256	66436.0	66411.0	**66367.8**	3500
Tai80b	818415043	826026281	821885368	**82000985**	30000
Sko100a	152002	152214	152135.2	**152058.8**	30000
Tho150	8133398	8151250	8142022.0	**8141724.0**	50000

如, 对 Tai30b 实例, FANT 和 ABFANT 算法是迭代 350 次, 而 DPSO 算法是迭代 70 次, 这样, 它们都调用了 700 次局部搜索算子. 表中数据除了 Tho150 实例是 5 次运行的平均值外, 其它实例的结果都是 20 次运行结果的平均值. 在所使用的 17 个 QAP 实例中对三种算法进行比较 (表中黑体为三个算法中的最好结果), 从表中可以看出, FANT 算法只在 Tai30b 实例上性能最好, 而 ABFANT 算法在其中 3 个实例 (Chr22b, Esc32a 和 Ste36b) 上性能最好, 在其余 13 个实例中, DPSO 算法都具有最佳的性能.

7 结论

以求解 QAP 问题为例, 提出了一个 DPSO 算法的具体实现, 算法中使用了排斥算子来保持粒子群的多样性, 采用了局部搜索算子来提高算法的局部求精能力, 使算法在空间勘探和局部求精间取得了较好的平衡, 仿真实验表明 DPSO 算法具有良好的性能, 说明如果能够针对问题的特点, 重新定义 DPSO 算法的位置、速度等量及其运算规则, DPSO 算法能够很好地应用到 NP- 困难的组合优化问题的求解中.

References

1 Kennedy J, Eberhart R. Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks. IEEE, 1995. 1942∼1948

2 Eberhart R, Kennedy J. A new optimizer using particle swarm theory. In: Proceedings of IEEE Sixth International Symposium on Micro Machine and Human Science. IEEE, 1995. 39∼43

3 Kennedy J, Eberhart R. A discrete binary version of the particle swarm algorithm. In: Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics. IEEE, 1997. 4104∼4109

4 Cagnina L, Esquivel S, Gallard R. Particle swarm optimization for sequencing problems: a case study. In: Proceedings of the 2004 Congress on Evolutionary Computation. IEEE, 2004, **1**: 536∼541

5 Tasgetiren M F, Sevkli M, Liang Y C, Gencyilmaz G. Particle swarm optimization algorithm for permutation flow shop sequencing problem. In: Proceedings of the 4th International Workshop on Ant Colony Optimization and Swarm Intelligence. Berlin: Springer-Verlag, 2004. 382∼390

6 Tasgetiren M F, Sevkli M, Liang Y C, Gencyilmaz G. Particle swarm optimization algorithm for single machine total weighted tardiness problem. In: Proceedings of the 2004 Congress on Evolutionary Computation. IEEE, 2004. 1412∼1419

7 Li Ning, Zou Tong, Sun De-Bao. Particle swarm optimization for vehicle routing problem with time windows. *System Engineering Theory and Practice*, 2004, **24**(4): 130∼135
(李宁, 邹彤, 孙德宝. 带时间窗车辆路径问题的粒子群算法. 系统工程理论与实践, 2004, **24**(4): 130∼135)

8 Taillard E D. FANT: Fast Ant System, Technical Report IDSIA-46-98, Lugano: IDSIA, 1998

9 Taillard E D, Gambardella L. Adaptive Memories for the Quadratic Assignment Problems, Technical Report IDSIA-87-97, Lugano: IDSIA, 1997

10 Zou Peng, Zhou Zhi, Chen Guo-Liang, Jiang He, Gu Jun. Approximate backbone guided fast ant algorithms to QAP. *Journal of Software*, 2005, **16**(10): 1691∼1698
(邹鹏, 周智, 陈国良, 江贺, 顾钧. 求解 QAP 问题的近似骨架导向快速蚁群算法, 软件学报, 2005, **16**(10): 1691∼1698)

11 Clerc M. Discrete particle swarm optimization. *New Optimization Techniques in Engineering*. Berlin: Springer-Verlag, 2004. 219∼240

12 Burkard R E, Karisch S, Rendl F. QAPLIB - A quadratic assignment problem library. *European Journal of Operational Research*, 1991, **55**(1): 115∼119

钟一文 副教授. 主要研究方向为计算智能、优化和并行算法. 本文通信作者. E-mail: yiwenzhong@163.com
(**ZHONG Yi-Wen** Associate professor. His research interest covers computational intelligence, optimization, and parallel algorithm. Corresponding author of this paper.)

蔡荣英 高级教师. 主要研究方向为运筹与优化.
E-mail: rongyingcai@163.com
(**CAI Rong-Ying** Senior teacher. Her research interest covers operational research and optimization.)