文章编号:1006-9348(2004)07-0110-05

蚁群算法及其实现方法研究

胡娟,王常青,韩伟,全智
(中国科学院软件研究所,北京 100080)

摘要:蚁群算法是一种相对较新的启发式方法,通过模拟蚂蚁的觅食行为解决问题,是目前昆虫算法中较成功的例子。蚁群算法的本质是一种并行的、自组织的算法,它可应用于更好地组织大数目实体的相互作用过程,如货郎担问题、车辆绕径问题、排程问题等。该文简述了蚁群算法的起源和发展,总结了蚁群算法的特点和不足及针对这些不足提出的各种改进方法,并介绍了和蚁群算法相关的几种具体应用。最后,文章探讨了蚁群算法研究中仍存在的问题和以后的发展方向。

关键词:蚁群;启发式;局部搜索
中图分类号:TP18 **文献标识码**:A

Research on the Ant Colony Optimization and Its Implementation Strategy

HU Juan, WANG Chang-qing, HAN Wei, HE Hui, QUAN Zhi
(Institute of Software, Chinese Academy of Sciences, Beijing 100080, China)

ABSTRACT: The Ant Colony Optimization(ACO) is a relatively new meta-heuristic algorithm and a successful paradigm of all the algorithms which take advantage of the insects' behavior. The ACO solves problems through mimicking ants' foraging behavior. Essentially, the ACO is a parallel and self-organizing algorithm, which can be applied to improve the management and control of large numbers of interacting entities such as TSP(Travelling Salesman Problem), Vehicle Routing Problem and Scheduling Problems. This article presents the origin and enrichment of the ACO, summarizes this algorithm's advantages, disadvantages, methods to overcome these disadvantages, and introduces some applications of the ACO. After discussing several problems existing in the research, this article puts forward the research foreground of ACO.

KEYWORDS: Ant system; Heuristic; Local search

1 引言

动物界的智能行为一直是科学家灵感的源泉,近年来,群居的昆虫表现出来的集体智慧吸引了研究者的注意。1979年,Dougls R. Hofstadter首次提出了人工蚂蚁的概念[1],探讨较低智能的个体间能否通过相互作用而产生较高的智能。从此,蚁群的自组织特性受到越来越多的关注。

2 AS和ACO算法的提出

1989年,Goss等通过著名的双桥实验[2]对阿根廷蚂蚁的觅食行为进行了研究。如图1所示,建立从巢穴到食物的两条路径:OAB和OB。两条路径均是双向连通的,且从点O到B只有这两条路径。一段时间后,蚁群中大部分蚂蚁都选择了较短路径OB。另外,实验还证明:蚂蚁选择较短路径的可能性和两条路径的距离差有关,随着距离差距增加,选择较短路径的可能性也在增加。

信息素遗留和跟踪理论对蚂蚁的这种行为作出了解释[3][4]:

1)每只蚂蚁行走过程中都会在自己走过的道路上留下一定数量的信息素;

2)信息素是一种易于挥发、可累加的化学物质;

3)道路上的信息素浓度越大,蚂蚁选择这条道路的可能性就越大。

在初始状态,蚂蚁随机地选择一条道路。由于相同时间内,较短的路径上会积聚较多的信息素,因而后面出发的蚂蚁倾向于选择路径OB——这样又进一步增加了OB上的信息素。一段时间后,大部分蚂蚁都选择了道路OB。

2.1 AS算法

受蚂蚁行为的启发,Colorni和Dorigo等人于1992年提出Ant System(AS)的概念[5][6]。AS算法最初包括ant-cycle、ant-density和ant-quantity三种算法,由于实验结果表明另两种算法效果不如ant-cycle,因此,主要在算法ant-cycle的基础上产生了Ant System。研究成果经应用于传统的货郎担问题(Travelling Salesman Problem——TSP)上,取得了很好的效果[7]。

算法1:Ant System for TSP

基金项目:中国科学院知识创新工程方向性研究课题资助项目(KGCX2-JG-09)
收稿日期:2004-03-31

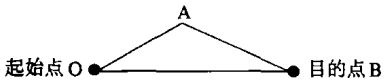

图1 蚂蚁觅食试验示意图

```
Initialize Graph;
while (terminate-condition-not-satisfied)
    for m = 1 to Number-of-ants
        while (not-every-city-visited)
            select-next-city;
        end while;
        Calculate-tour-length;
    end for;
    Update-pheromone-of-graph;
end while;
```

算法首先初始化图结构，定义 TSP 问题的城市数量、连通情况、路径长度和信息素的初始值；terminate-condition-not-satisfied：算法的结束条件未能满足（结束条件包括：循环次数达到最大值、解在一定次数循环内没有改进、算法已经取得最优解等）；select-next-city：使用算法定义的路径选择规则(1)选择蚂蚁下一步的走法；Calculate-tour-length：计算蚂蚁走过的路线的长度；Update-pheromone-of-graph：更新信息素，使用信息素的更新规则(2)。

$$p_{ij}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha} \times [\eta_{ij}(t)]^{\beta}}{\sum_{k \in allowed\ nodes} [\tau_{ik}(t)]^{\alpha} \times [\eta_{ik}(t)]^{\beta}} & if\ j \in allowed\ nodes \\ 0 & 否则 \end{cases} \tag{1}$$

其中，$\eta_{ij} = \frac{1}{d_{ij}}$ 表示路径的启发信息；α 和 β 是决定信息素和启发信息的权，需要通过手工调节；τ 是信息素数量。

$$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \Delta\tau_{ij}(t) \tag{2}$$

其中，ρ 是信息素的挥发率，$\Delta\tau_{ij}$ 是信息素的增加，计算方法是：

$$\Delta\tau_{ij}(t) = \sum_{m=1}^{Number\text{-}of\text{-}Ants} \frac{Q}{L_m} \quad if\ arc(i,j) \in Tour\text{-}of\text{-}Ant_m \tag{3}$$

Q 是一个固定的正数；L_m 是第 m 只蚂蚁行走的路线长度。

2.2 ACO 算法

1998 年，Dorigo 和 Di Cargo 进一步发展了 AS 算法，强调了启发信息的重要性并使用了多种方法提高算法性能，形成 ACO 算法[8]。目前，ACO 算法已被应用于解决大量问题如：货郎担问题[9]、车辆绕径问题(Vehicle Routing Problem)[10][11]、排程问题(Scheduling Problems)[12-16]等。

ACO 算法具有区别于以往优化算法的特点：①通用性：ACO 适合解决能转换成为连通图结构的问题，包括许多 NP 难题，此外还能解决图结构属性在计算过程中发生变化的问题。②正反馈：使用局部解来构造全局解，通常是加强局部的较优解。③负反馈：通过信息素挥发避免算法陷入局部收敛。④间接通讯：蚂蚁之间没有直接的联系，信息的传递是通过路径上的信息素。⑤自组织：ACO 依靠群体的力量解决问题。

使用 ACO 解决类似 TSP 问题的算法如下：

算法 2：ACO 算法

```
Initialize Graph;
while (terminate-condition-not-satisfied)
    for m = 1 to Number-of-ants
        while (not-every-city-visited)
            read-ant-routing-table;
            select-next-city;
            If(online-step - by - step-pheromone-update)
                Deposit-pheromone-on-the-visited-arc();
                Update-ant-routing-table();
            End-if;
        end while;
        Calculate-tour-length;
        If (online-delayed-pheromone-update)
            For each visited-arc do
                Deposit-pheromone-on-the-visited-arc();
                Update-ant-routing-table();
            End-for;
        End-if;
    end for;
    Update-pheromone-of-graph;
    daemon-actions(); {optional}
end while;
```

相对于算法 1，ACO 使用了 ant-routing-table 来记录蚂蚁行走的过程。当蚂蚁从一个点移动到另一点时，可随时改变信息素，称为在线更新信息素（Online step - by - step pheromone update）；一旦得到一个解，蚂蚁能够回溯这个解（更新信息素，称为在线定时更新：Online delayed pheromone update）。挥发避免了算法提前收敛于次优解；后台处理集中控制算法的收敛如从蚂蚁的路径中选择部分，加以特别的信息素（这种信息素的更新称为离线更新——Offline pheromone update）等。

文献[17]从理论上证明了群体智能（Swarm Intelligence）是符合统计规律的。

3 对 ACO 算法的改进

ACO 算法提出后，吸引了研究者的注意。文献[18]分析了算法中存在的问题：蚂蚁在走第一步时依照信息素的多少确定概率，但在第二步中却没将第一步的概率考虑在内，因而造成了偏差。文献[18]提出了相应的解决方法并在 Single Machine Total Weighted Deviation Problem(SMTWDP)问题中验证了相应的改进。

3.1 信息素的更新

文献[19]中，不是每一代都对最好的解更新信息素，而是过了一定数量的代后再更新。在每一代，都根据一个较小的概率，在本代得到的最优解与目前为止算法的最优解中取优。信息素的表达方式：文献[20]中比较了四种信息素的表达方式(式(4)至(7))在解答 FOP 问题时的效率。

$$\begin{cases} \dfrac{\tau_{O_j,t}}{\sum_{O_k \in J_t} \tau_{O_k,t}} & if\ O_j \in J_t \\ 0 & 否则 \end{cases} \tag{4}$$

$$\begin{cases} \dfrac{\sum_{l=1}^{t} \tau_{O_j,l}}{\sum_{O_k \in J_t} \sum_{l=1}^{t} \tau_{O_k,l}} & if\ O_j \in J_t \\ 0 & 否则 \end{cases} \tag{5}$$

$$\begin{cases} \dfrac{\tau_{V_{Source},O_j}}{\sum_{O_k \in J_t} \tau_{V_{Source},O_k}} & if\ O_j \in J_t, t = 1 \\ \dfrac{\tau_{O_i,O_j}}{\sum_{O_k \in J_t} \tau_{O_i,O_k}} & if\ O_j \in J_t, t > 1, s[t-1] = O_i \\ 0 & 否则 \end{cases} \tag{6}$$

$$\begin{cases} \dfrac{\min_{O_r \in S_{O_j}^{rel}} \tau_{O_j,O_r}}{\sum_{O_k \in J_t} \min_{O_r \in S_{O_k}^{rel}} \tau_{O_k,O_r}} & if\ O_j \in J_t \\ 0 & 否则 \end{cases} \tag{7}$$

实验证明，在使用临接图结构表示问题时，式(7)具有明显的优势[20]。

3.2 适应函数的计算

文献[21]中，提出了新的计算方法：

$$F = \sum_{i=1}^{4} C_i \cdot F_i \tag{8}$$

1) $F_1 = \dfrac{1}{L_{+1} - L_+}$，$C_1 = 2.0 - 3.0$。$L_{+1}$ 是本代蚂蚁找到的最好解，L_+ 是前面找到的所有解里面最好的。

2) $F_2 = e^{-\frac{V}{5 \cdot n}}$，$C_2 = 0.5 - 0.8$，$V$ 是找到最好解的蚂蚁代数，表示找到最好解的程度。

3) $F_3 = e^{-\frac{m}{10 \cdot n}}$，$C_3 = 0.5$，尽量减少蚂蚁数目 m。

4) $F_4 = e^{-\frac{g}{n}}$，$C_4 = 0.2 - 0.5$，尽量减少交换次数。

另一种方法是：在同一个蚁群中，分出来几个使用不同参数的小组。用此方法曾在 Oliver30 问题上取得最好的值 70ant - cycles(7cycles - 10ants)，但这种方法还缺乏大量验证。

文献提出了一种综合评价策略，综合考虑本次循环得到的解相比较于最优解的差距和优化的程度，应用于 JSP 问题中，取得了很好的效果：

$$F_{evaluate} = \omega \times e^{1 - f_{current}/Best\text{-}Value} + \theta \times [e^{(f_{prior} - f_{current})/f_{prior}} - 1] \tag{9}$$

其中，第一部分：$e^{1 - f_{current}/Best\text{-}Value}$ 表示当前解和最优解的差距；第二部分：$e^{(f_{prior} - f_{current})/f_{prior}} - 1$ 表示本次求得的解比较上一次的优化程度；ω 和 θ 是固定系数，表征两个部分在函数中所占有的比例；$f_{current}$ 表示当前的蚂蚁路径代表的 JSSP 问题中最小时间的值；f_{prior} 表示上一代蚂蚁的最小时间的最优值。

3.3 与局部搜索的结合

ACO 与 GA 的结合使用源于这种考虑：ACO 算法中的参数需要手工确定，且常与问题的规模有关，使用 GA 确定 ACO 的参数便能得到合适的值，从而提高算法的性能[21][23]。文献[21]中使用这种方法明显提高了算法的收敛速度，在某些情况下还能取得非常优秀的解。局部搜索往往还直接与 ACO 相结合。使用局部搜索将蚂蚁找到的局部最优解重新组合以产生新的更优化的解。

文献[22]中提出了在局部搜索中可能用到的三种方法：

1) swaps of two neighboring jobs at position i and i + 1, I = 1, 2...n - 1(swap - moves)

2) exchanges of jobs placed at the I th and the j th position, i ≠ j(interchange - moves)

3) remove the job at the I th position and insert it in the j th position (insertion - moves)

基于 swap - move 的局部搜索不能得到足够数量的满意解，在文献[23][24]中，作者指出，在基于临接结构的问题中，insertion - moves 比 interchange - moves 更有效，至少不比后者差。同时，采用预处理程序加快了算法的收敛速度。

此外，文献[25]使用了一种 ACO 和禁忌搜寻法(Tabu Search)相结合的方法解决 Group Shop 排程问题。

4 基于具体应用的 ACO 算法的发展

为提高 ACO 算法的效率，研究人员从许多方面着手改进算法，主要有：算法信息素的表达方式和更新方式方面的改进；路径选择策略方面的改进；与其他算法的结合；使用预处理程序提高算法性能；适应函数计算方法的改进等。比较成功的研究成果有：ASrank 算法、ACS 算法、MMAS 算法、Ant - Solver 算法等。

4.1 The Rank - based Version of Ant System(AS_{rank})[26]

AS_{rank} 算法可以看作精英策略的推广。

精英策略(elitist strategy[27])是应用在算法的信息素更新方面的，首先由 Dorigo 等人提出。算法记录历史上最好的解，每次更新信息素时，不是针对本次循环得到的最好的解做信息素的增加，而是更新全局最优解所包含的路径。应用于 TSP 算法的更新规则是式(10)。

$$\tau_{ij}(t+1) = (1 - \rho) \cdot \tau_{ij}(t) + \Delta\tau_{ij}(t) + \Delta\tau_{ij}^{*}(t) \tag{10}$$

其中，$\Delta\tau_{ij}(t)$ 的计算仍使用式(3)，$\Delta\tau_{ij}^{*}(t)$ 的计算使用式(11)。

$$\Delta\tau_{ij}^{*}(t) = \begin{cases} \dfrac{k}{L(t)} & if\ arc(i,j) \in T^{global\text{-}best} \\ 0 & 否则 \end{cases} \tag{11}$$

如果连接点 i、j 的边是全局最优解的一部分的话，在每次更新信息素时就对这条边增加信息素。增加的量与全局最优解的大小有关。k 是一个正数。

AS_{rank} 算法取得每次循环中最好的($\sigma-1$)个最优解和当前的全局最优解，并对这些解相应的路径更新信息素。更新的规则仍旧是式(10)，只是其中的计算不再使用规则(3)而是使用规则(12)。

$$\Delta\tau^{ij} = \sum_{\upsilon=1}^{\sigma-1}\Delta\tau_{ij}^{\upsilon} \tag{12}$$

其中 $\Delta\tau_{ij}^{\upsilon}$ 的计算使用规则(13)。

$$\Delta\tau_{ij}^{\upsilon} = \begin{cases}(\sigma-\upsilon)\cdot\dfrac{Q}{L_{\upsilon}} if\ arc(i,j)\in T^{the\text{-}\upsilon\text{-}th\text{-}best\text{-}tour}\\ 0 \qquad\qquad 否则\end{cases} \tag{13}$$

4.2 ACS 算法

相对于原始的 AS 算法，ACS 算法在三个方面进行了改进：

1) 信息素的更新使用精英策略，针对每一次循环得到的最优解和算法的全局最优解应用规则(14)[28]。

$$\tau_{ij}(t+1) = (1-\rho)\cdot\tau_{ij}(t) + \rho\cdot\Delta\tau_{ij}^{best}(t) \tag{14}$$

2) 选择下一条路径的可能性是根据伪随机比例原则确定的：

$$p_{ij}(t) = \begin{cases}\dfrac{\tau_{ij}(t)\times\eta_{ij}(t)^{\beta}}{\sum_{k\in allowed\ nodes}[\tau_{ik}(t)]\times[\eta_{ik}(t)]^{\beta}} if\ j\in allowed\ nodes\\ 0 \qquad\qquad 否则\end{cases} \tag{15}$$

$$p' = \begin{cases}\arg\ Max\{[\tau(i,j)]\cdot[\eta(i,j)]^{\beta}\}_{k\in allowed\ nodes} & if\ q\leqslant q_0\\ p_{ij}(t) \qquad 否则\end{cases} \tag{16}$$

其中，q 是一个随机数，q_0 是一个变量，表征规则探索新的路径(exploration)还是利用已有的路径(exploitation)。若 $q\leqslant q_0$，算法开始探索新的路径；否则，算法使用已有路径的信息决定下一步的概率。

3) 使用本地信息素更新的方法。除全局更新外，算法在每一个蚂蚁循环中对信息素也作更新。

4.3 MAX - MIN Ant System[12][29]

MAX - MIN Ant System(MMAS)算法使用一个信息素上限 τ_{max} 和一个信息素下限 τ_{min}。路径上信息素的初始值设置为 τ_{max}。信息素下限 τ_{min} 的作用是阻止算法收敛。算法在计算过程中，每一代蚂蚁行走结束后都奖励迄今为止的全局最优解。由于 MMAS 算法加强了信息素的更新，算法会很快收敛于某点，然后更新信息素，重新开始计算。

MMAS 算法也常和局部搜索结合以提高性能。

4.4 Ant - Solver[22][30][31]

Ant - Solver 算法是专门为解决 Constraint Satisfaction Problem(CSP)而提出的一种类 AS 算法。它将 AS 算法适当简化，使用一个随机构造 CSP 的解的过程代替 AS 中蚂蚁的行走，在构造结束后使用构造的结果对信息素进行更新。信息素的更新遵循规则(17)。

$$\tau_{ij} = (1-\rho)\cdot\tau_{ij} + \sum_{A_k\in BestOfCycle}\Delta\tau_{ij}(A_k) \tag{17}$$

Ant - Solver 算法常和局部搜索相结合，也往往加入预处理程序以提高性能。文献[31]中使用了最小冲突启发算法作为局部搜索方法。

5 结论

本文对 ACO 算法进行了综述，介绍了算法发展过程及各阶段对算法的改进。由于 ACO 算法为人们提供了一种新的自组织的思路，通过个体的无意识行为的组合而得到群体的较高智能的行为，因而可应用于很多组合优化的场合(如机器人的协同工作[32]等)。

关于 ACO 算法，目前还存在着许多问题有待研究：首先，ACO 算法虽具有较低的空间复杂度，但时间复杂度较高(如，ACO 中每一次循环都针对每一只蚂蚁求解问题)，虽然人们使用 ACO 与其他方法结合来解决这个问题，但是这些组合的方法在应用于规模较大的问题时仍显不足；其次，ACO 能够解决的问题的范围仍有待扩展，目前算法能够解决的问题只局限于定义良好的图结构，对于图结构的属性在解答过程中发生变化的问题无能为力，如何将算法应用于这类问题中将是一个重要的方向；最后，对于算法本身的改进，目前的研究工作是针对算法中不同的部分做修改，对各种方法的综合使用及各种改进方法的相互作用的研究还没有涉及，这也是以后研究的重点之一。

参考文献：

[1] D R Hofstadter. G? del, Escher, Bach: an Eternal Golden Braid [M]. Basic Books, New York, 1979.

[2] S. Goss, S. Aron, J. L. Deneubourg, and J. M. Pasteels. Self - Organized shortcuts in the Argentine ant[J]. Naturwissenschaften, 1989, 76: 579 - 581.

[3] V A Cicirello and S F Smith. Insect Societies and Manufacturing[R]. In IJCAI - 01 Workshop on Artificial Intelligence and Manufacturing: New AI Paradigms for Manufacturing, August 2001.

[4] B H? lldobler and E O Wilson. The Ants[R]. The Belknap Press of Harvard University Press, 1990.

[5] A Colorni, M Dorigo, and V Maniezzo. Distributed optimization by ant colonies[C]. In Proceedings of the First European Conference on Artificial Life, pages 134 - 142. Elsevier Publishing, Paris, France 1992.

[6] A Colorni, M Dorigo, and V Maniezzo. An investigation of some properties of an "ant algorithm"[C]. In Proceedings of the Parallel Problem Solving from Nature Conference, pages 509 - 520. Elsevier Publishing, Brussels, Belgium 1992.

[7] M Dorigo, V Maniezzo, and A Colorni. Ant system: Optimization by a colony of cooperating agents[J]. IEEE Transactions on Systems, Man,

and Cybernetics - Part B: Cybernetics, 26(1):29 - 41, February 1996. From Animals to Animats: Proceedings of the First International.

[8] M Dorigo, G Di Cargo and L M Gambardella. Ant algorithms for distributed discrete optimization[R]. Technical Report 98 - 10 IRIDIA, Université Libre de Brexelles, 1998.

[9] M Dorigo and G Di Caro. The ant colony optimization meta - heuristic. In D. Corne, M. Dorigo, and F. Glover, editors, New Ideas in Optimization. McGraw - Hill, 1999.

[10] B Bullnheimer, R F Hartl, and C Strauss. An improved ant system algorithm for the vehicle routing problem[J]. Annals of Operations Research, 89:319 - 328, 1999.

[11] L M Gambardella, é Taillard and G Agazzi. MACS - VRPTW: A multiple ant colony system for vehicle routing problems with time windows[R]. Technical Report IDSIA - 06 - 99, Istituto Dalle Molle di Studi sull' Intelligenza Artificiale (IDSIA), Lugano, Switzerland, 1999.

[12] A Colorni, M Dorigo, V Maniezzo and M Trubian. Ant System for Job - Shop Scheduling[J]. Belgian Journal of Operations Research, Statistics and Computer Science, 1994,34(1):39 - 53.

[13] S van der Zwaan and C Marques. Ant colony optimization for job shop scheduling[C]. In Proceedings of the '99 Workshop on Genetic Algorithms and Artificial Life GAAL'99, Lisbon, Portugal, March 1999.

[14] P Forsyth and A Wren. An ant system for bus driver scheduling[R]. Technical Report 97.25, University of Leeds, School of Computer Studies, July 1997. Presented at the 7th International Workshop on Computer - Aided Scheduling of Public Transport, Boston, July 1997.

[15] M den Besten, T St¨utzle and M Dorigo. Ant colony optimization for the total weighted tardiness problem[C]. In Proceedings of the Parallel Problem Solving from Nature Conference, 2000.

[16] D Merkle, M Middendorf and H Schmeck. Ant colony optimization for resource - constrained project scheduling[C]. In GECCO - 2000: Proceedings of the Genetic and Evolutionary Computation Conference, pages 893 - 900. Morgan Kaufmann, July 2000.

[17] Dima Iourinski, Scott A Starks, Vladik Kreinovich. Swarm Intelligence: Theoretical Proof That Empirical Techniques Are Optimal[C]. Proceedings of the 2002 World Automation Congress WAC'2002, Orlando, Florida, June 9 - 13, 2002. 107 - 112.

[18] Daniel Merkle, Martin Middendorf . A New Approach to Solve Permutation Scheduling Problems with Ant Colony Optimization - 2001[C]. In E.J.W. Boers et al. (Eds.) Applications of Evolutionary Computing: Proceedings of EvoWorkshops 2001, Lake Como, Italy, Springer Verlag, LNCS 2037,2001.484 - 493.

[19] D Merkle, M Middendorf and H Schmeck. Ant colony optimization for resource - constrained project scheduling[C]. In GECCO - 2000: Proceedings of the Genetic and Evolutionary Computation Conference, pages 893 - 900. Morgan Kaufmann, July 2000.

[20] Christian Blum, Michael Sampels. Ant Colony Optimization for FOP Shop Scheduling: A case study on different pheromone representations [C]. In Proceedings of the 2002 Congress on Evolutionary Computation, CEC'02. 1558 - 1563.

[21] Hozefa M Botee, Eric Bonabeau. Evolving Ant Colony Optimization [C]. In Advances in Complex Systems, 1999, 1(2/3):149 - 159.

[22] Christine Solnon Boosting ACO with a Preprocessing Step[C]. In Applications of Evolutionary Computing, Proceedings of EvoWorkshops2002: EvoCOP, EvoIASP, EvoSTim.

[23] E Taillard. Some Efficient Heuristic Methods for the Flow Shop Sequencing Problem[J]. European Journal of Operational Research, 1990,47:65 - 74.

[24] I Osman and C Potts. Simulated Annealing for Permutation Flow - Shop Scheduling[J]. OMEGA, 1989,17(6):551 - 557.

[25] Christian Blum. ACO applied to Group Shop Scheduling: A case study on Intensification and Diversification[C]. In M. Dorigo, G. Di Caro, and M. Sampels, editors, Proceedings of ANTS 2002 - Third International Workshop on Ant Algorithms, volume 2463 of Lecture Notes in Computer Science, Springer Verlag, Berlin, Germany, 2002. 14 - 27.9

[26] B Bullnheimer, R F Hartl and C Strauss. A New Rank Based Version of the Ant System - A Computational Study[R]. Technical report, University of Viena, Institute of Management Science, 1997.

[27] M Dorigo, V Maniezzo and A Colorni. The Ant System: Optimization by a colony of cooperating agents[J]. IEEE Transactions on Systems, Man, and Cybernetics - Part B, 1996, 26(1):29 - 41.

[28] M Dorigo and L M Gambardella. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem[J]. IEEE Transactions on Evolutionary Computation, 1997, 1(1): 53 - 66.

[29] T Stützle and H Hoos. Improvements on the Ant System: Introducing MAX - MIN Ant System[M]. In G. D. Smith, N. C. Steele, R. A., editor, Artificial Neural Networks and Genetic Algorithms, 1998.245 - 249.

[30] C Solnon. Solving permutation constraint satisfaction problems with artificial ants[C]. In W. Horn, editor, Proceedings of the 14th European Conference on Artificial Intelligence, pages 118 - 122. IOS Press, Amsterdam, The Netherlands, 2000.

[31] Christine Solnon. Ants Can Solve Constraint Satisfaction Problems[J]. in IEEE Transactions on Evolutionary Computation, August 2002,6(4):347 - 357.

[32] J L Deneubourg, S Goss, N Franks, A Sendova - Franks, C Detrain and L Chr'etien. The dynamics of collective sorting robot - like ants and ant - like robots[C]. In From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior, MIT Press, 1991. 356 - 363.

[作者简介]

胡娟(1977.7 -),女(汉族),湖南沅江人,硕士研究生,主要研究方向为并行计算和人工智能;

王常青(1978.5 -),男(汉族),山东青岛人,博士研究生,主要研究方向为并行计算和人工智能;

韩伟(1977.12 -),男(汉族),山东青岛人,硕士研究生,主要研究方向为并行计算和人工智能;

全智(1978.6 -),女(汉族),湖南衡阳人,硕士研究生,主要研究方向为并行计算和神经网络。