Software Engineering and Applications 软件工程与应用, 2020, 9(6), 467-474
Published Online December 2020 in Hans. http://www.hanspub.org/journal/sea
https://doi.org/10.12677/sea.2020.96054

Hans汉斯

一种基于子问题动态消减的改进多目标蚁群优化算法

宁佳绪，牛　玥，纪丹蕾，肖雨婷，杨富燕

沈阳理工大学，信息科学与工程学院，辽宁 沈阳
Email: 739250969@qq.com

收稿日期：2020年11月16日；录用日期：2020年12月14日；发布日期：2020年12月21日

摘　要

为进一步提高基于分解的多目标蚁群优化算法的性能，提出了一种子问题动态消减方法并将其结合到MOEA/D-ACO算法中，以此提出了一种基于子问题动态消减的改进多目标蚁群优化算法IMOEA/D-ACO。该算法通过在运行早期识别没有前途的子问题并及时抛弃对其进行搜索来提高搜索资源的利用率。从而在搜索资源总量一定的前提下，能够进一步提升算法的性能。为了验证算法性能分别与其他相关算法在不同规模的TSP问题测试用例上进行了实验比较。结果表明IMOEA/D-ACO算法在求解质量上优于被比较算法。

关键词

群智能，多目标优化，蚁群优化算法，信息素，支配

An Improved Multi-Objective Ant Colony Optimization Algorithm Based on Sub-Problems Dynamic Subtraction

Jiaxu Ning, Yue Niu, Danlei Ji, Yuting Xiao, Fuyang Yang

College of Information Science and Engineering, Shenyang Ligong University, Shenyang Liaoning
Email: 739250969@qq.com

Received: Nov. 16th, 2020; accepted: Dec. 14th, 2020; published: Dec. 21th, 2020

Abstract

To further improve the performance of decomposition based multi-objective ant colony algorithm, a dynamic sub-problem reduction method is proposed and combined with the MOEA/D-ACO algorithm. Based on this, a sub-problem dynamic reduction improved multi-objective ant colony algo-

文章引用: 宁佳绪, 牛玥, 纪丹蕾, 肖雨婷, 杨富燕. 一种基于子问题动态消减的改进多目标蚁群优化算法[J]. 软件工程与应用, 2020, 9(6): 467-474. DOI: 10.12677/sea.2020.96054

rithm called IMOEA/D-ACO is designed. Through identifying the unpromising sub-problems during the early optimizing process and giving them up in time for optimizing, the utilization of the searching resource is further increased. Thus the algorithm performance can be improved when the total consumed resources are fixed. To verify its performance, it is tested on some TSP instances with different scale, and compared with some related algorithms. The results show that the proposed algorithm is superior to the compared algorithms.

Keywords

Swarm Intelligence, Multi-Objective Optimization, Ant Colony Optimization Algorithm, Pheromone, Dominance

1. 引言

近年来，在解决多目标优化问题尤其是路径优化问题时，多目标蚁群算法(multi-objective ant colony optimization algorithms, MOACOs)相对于其他元启发式算法被更广泛采用[1] [2] [3]，因此对其进行研究具有一定价值。

目前多目标蚁群算法可分为基于非支配排序和基于分解两类。基于非支配排序的多目标蚁群算法产生解的个数比较多，但分布不均匀，通常集中在某一方向上。根据使用信息素矩阵的个数的不同，这类算法又可分为单信息素矩阵多目标蚁群优化算法如 MOAQ [4]和 MACS 算法[5]，以及多信息素矩阵多目标蚁群优化算法如 PACO [6]、BicriterionAnt [7]和 MOACO 算法[8]。2015 年 I. D. I. D Ariyasingha 等[9]对 2012 年之前的 7 个基于非支配排序的多目标蚁群算法在多个 TSP 测试用例上进行了详细的对比分析。Liangjun Ke 等[10]结合了多目标进化算法(EA)，提出了基于进化分解的蚁群算法(MOEA/D-ACO)，该方法中每个子问题对应一个启发式信息和一个历史最优解，将蚂蚁分为若干组，每只蚂蚁只能分配在一个群组中，每组蚂蚁有自己的信息素矩阵，每只蚂蚁解决一个子问题，并通过聚合它所属组的信息素、子问题的启发式信息和历史最优解来构造新解。2018 年作者[11]提出了基于分解的多目标蚁群优化算法 MOACO/D-ACS，将一个多目标问题分解为多个单目标子问题，蚂蚁分为多个子群，每个子群有各自的信息素矩阵和启发式矩阵，每只蚂蚁可以属于多个子群，也就是说算法每次迭代，在不同组中的同一只蚂蚁都会产生一个解，并用其构造的解更新所在子群的信息素矩阵。

基于分解的多目标蚁群算法获得的帕累托前沿分布相对比较均匀，而且求解质量和效率都好于已有的基于支配的多目标蚁群优化算法。在已有的基于分解的多目标蚁群算法中，在初始化阶段普遍采用切比雪夫方法对多目标优化问题进行分解，通过生成 N 个权重向量 $\left(\lambda^1,\cdots,\lambda^n\right)$ 将一个多目标问题划分为 N 个单目标子问题，优化过程中每个子问题与一只蚂蚁相对应并且所有权重向量保持不变。当非支配解在搜索空间分布不均时，最优解不属于帕累托前沿的子问题数量会进一步增加。如果能够在算法优化过程中尽早识别并放弃针对这些子问题的求解能够进一步搜索资源的利用率，从而提高算法优化性能。为此，本文设计了一种子问题动态消减策略并将其结合到 MOEA/D-ACO 中，从而提出了改进算法 IMOEA/D-ACO。为验证其有效性，应用不同规模的测试问题对算法进行了测试并与多种相关算法进行了比较，结果表明，结合动态消减策略的 MOEA/D-ACO 算法不但收敛速度快而且大部分测试问题上求解质量更佳。

2. IMOEA/D-ACO 算法

IMOEA/D-ACO 是在 MOEA/D-ACO 算法的基础上改进子问题资源分配机制，通过设计的子问题动态削减策略在优化过程中逐步识别并淘汰一些没有前途的子问题，直到子问题的个数等于或小于资源数。以此通过提高资源利用率的方式进一步提升算法性能。

在 MOEA/D-ACO 算法中蚂蚁数和子问题数相同，每只蚂蚁固定地解决一个子问题，不存在为子问题分配蚂蚁的情况。由于并不清楚非支配解会出现在哪些子问题上，为尽可能多的识别出有前途的子问题，本文在进行子问题分解时生成的子问题数量 Q 大于群体中的蚂蚁数量 N。在蚂蚁构造解之前，首先要根据每个子问题的概率分配蚂蚁，并且算法前期迭代时，总会有一些子问题没有分配到蚂蚁，随着算法的运行，那些分配到蚂蚁的子问题如果连续几次没有产生非支配解，则将此子问题淘汰子问题，这样子问题的个数越来越接近于资源数，直到剩余的每个子问题每次迭代至少分配到一只蚂蚁。

为实现以上过程为每个子问题设置了两个属性：count1 和 count2。count1 的值用于记录子问题产生的非支配解的个数，每次产生一个非支配解就增加 1，以提高此子问题在下一次算法迭代时被蚂蚁选中的概率，而产生被支配解则保持不变。count2 的值用于记录子问题被选中后连续产生被支配解的次数，如果 count2 值大于给定的阈值 U 并且子问题的数量大于 N 的情况下将其直接淘汰，而当产生非支配解时则将其值重置为零。对于当前的未被淘汰的一个子问题 a，其被蚂蚁选择的概率计算公式如(1)所示：

$$P_a = \frac{count1_a}{\sum_{b\in B} count1_b} \quad \left(a \in \{1, 2, \cdots, Q\}\right) \tag{1}$$

其中，count1a 表示子问题 a 保存的 count1 的值；B 是当前子问题的候选集合，b 表示集合 B 中的一个未被访问的子问题。以此使得子问题产生的非支配解的个数越多即 count1 的值越大，该子问题下次迭代被蚂蚁选择的概率就越高。

基于以上描述 IMOEA/D-ACO 算法的基本过程如下：

Step 1：采用切比雪夫方法将带求解问题分解为 Q 个子问题，每个子问题 $i \in [1, Q]$ 对应的权重向量为 $\lambda_i = \left(\lambda_i^1, \cdots, \lambda_i^M\right)$，$M$ 为目标函数数量。

Step 2：初始化，与算法 MOEA/D-ACO 中 Step2 步骤一致。

Step 3：先判断子问题是否已经被淘汰。根据公式(3)计算每个子问题被选中的概率，为每只蚂蚁分配子问题，然后使用轮盘赌的方式为每个蚂蚁分配子问题。

Step 4：为每只蚂蚁构造路径，同 MOEA/D-ACO 中 Step3 步骤一致。

Step 5：对构建的路径进行评价，方法同算法 MOEA/D-ACO 中 Step4 步骤一致。

Step 6：根据每只蚂蚁产生的解是支配解还是非支配解更新子问题的参数 count1 和 count2 的值，并决定是否淘汰该子问题。

Step 7：利用蚂蚁构造的新解更新信息素矩阵。与 MOEA/D-ACO 算法中的 Step5 一致。

Step 8：更新每个子问题的历史最优解。与 MOEA/D-ACO 算法中的 Step6 一致。

Step 9：判断是否满足结束条件，满足则输出 Abf；否则，返回到 Step3 继续下一次迭代。

通过自适应的资源分配策略，算法 IMOEA/D-ACO 能够在更精细的子问题划分情况下，通过逐步淘汰不好的子问题即其对应的 PF 难以求解，将有限的资源尽可能多的应用到剩余的比较好的子问题上，有效的提高了资源利用率，在同等条件、资源等情况下，有很高的概率可以产生相对于原算法的更多更好的非支配解。

IMOEA/D-ACO 算法的复杂度分析：

设算法运行过程中产生的最多的子问题总数为 Q*ITER (其中 ITER 为迭代次数)，目标函数运算的时间复杂度为 $O(f)$，IMOEA/D-ACO 算法的时间复杂度为：

$$O(\text{IMOEA/D-ACO}) = (S * \text{ITER}) * O(f) + C_1$$

而 MOEA/D-ACO 算法的时间复杂度为：

$$O(\text{MOEA/D-ACO}) = (N * \text{ITER}) * O(f) + C_2$$

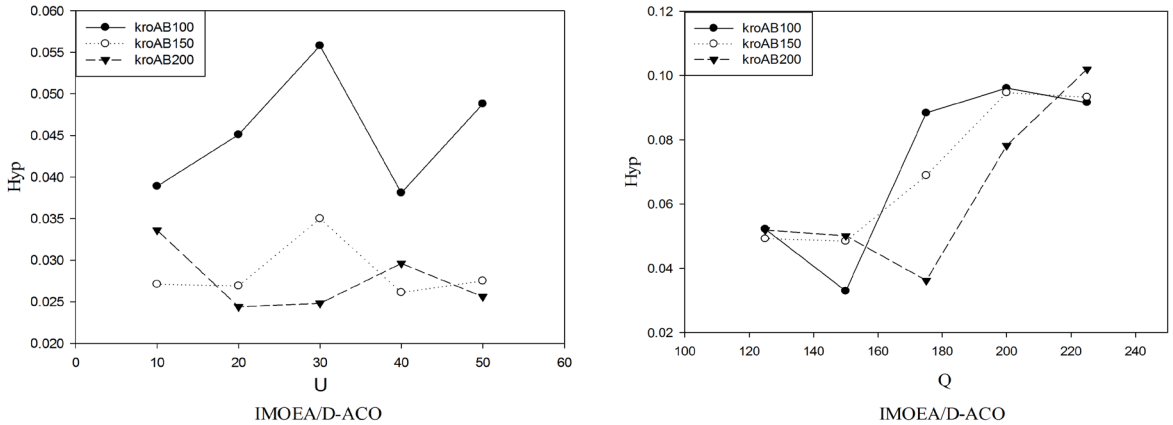
其中 C_1 为IMOEA/D-ACO 算法进行额外计算的复杂度，C_2 为MOEA/D-ACO 算法进行额外计算的复杂度，算法迭代次数为 ITER，则有 $C_1 = C_2 + \text{ITER} * O(Q)$，即 $C_1 > C_2$，又因为 $(S * \text{ITER}) * O(f) \gg \text{ITER} * O(Q)$ 且 S 与 N 相差为一个常量，其中 $S \le Q$ 为每次迭代过程中待求解的子问题数量。所以可以认为 $C_1 > C_2$ 对整体的运行时间不在一个量级上，可以忽略不计。即，改进后算法的时间复杂度与原算法相同。

3. 实验与结果

本节将通过在不同规模 TSP 问题上进行实验测试对 IMOEA/D-ACO 中的主要参数设置进行探讨，并将其与 MOAQ、MACS、PACO、BicriterionAnt、MOACO、MOACO/D-ACS、MOEA/D-ACO 等相关算法进行实验比较。实验中采用了 4 个标准的多目标 TSP 问题测试用例 kroAB100、kroAC100、kroAB150、kroAB200，可以从 http://eden.dei.uc.pt~paquete/tsp/下载。实验过程中各算法的终止条件为最大适应度评价次数：$N \times 300 \times (n/100)^2$，其中 N 代表蚂蚁的数量，n 代表城市的规模大小。被比较算法的其它参数均按原论文进行设置，IMOEA/D-ACO 算法的参数 Q 和 U 的设置将在 3.1 节进行探讨，其他参数设置与 MOEA/D-ACO 算法相同。实验过程中采用 Hypindicator (H-指标) [11]对算法性能进行比较。

3.1. 参数调整

本小节选择了 3 个不同规模的测试用例 kro-AB100、kro-AB150、kro-AB200 对 IMOEA/D-ACO 算法中参数 Q 和 U 的设置对算法性能的影响进行分析。为了保证测试的公平性，对每种参数值配置独立运行 30 次，将实验结果转化为 H-指标，然后取它们的平均值；H-指标默认越小越好。为了提高测试效率，采用固定一个参数值同时调整另一个参数的方式。IMOEA/D-ACO 算法中 Q 的取值范围为[125, 150, 175, 200, 225]，U 的范围为[30, 40, 50, 60, 70]。先将 Q 取中间数 175，然后调整 U 的取值。从图 1(a)中可以看出在测试用例 kroAB100 和 kroAB150 中，U 取 40 时，H 值最小；而对于测试用例 kroAB200，U 取 20 时 H 值最小。然后固定 U=40，调整组数 Q 的值。从图 1(b)可以看出，在测试用例 kroAB100 和 kroAB150 中，参数 Q=150 时 H 值最小；而对于测试用例 kroAB200，Q = 175 时 H 值最小。



(a) Parameter U　　(b) Parameter Q

Figure 1. The parameter adjustments of U and Q
图 1. 参数调整

3.2. 与相关算法比较

本节首先使用 H-指标来分析各个算法的性能，每个算法在各测试用例上独立运行 30 次，并将各个算法的实验结果转换成 H-指标。实验过程中在所有测试用例上参数 $U = 40$，$Q = 150$。

通过图 2 可以看出算法 IMOEA/D-ACO 在个测试问题上的求解质量明显好于其他被比较算法，并且在求解规模大小为 150 和 200 的两个测试问题上其优势更佳明显。通过把各个算法在每个问题上运行 30 次得出的非支配解集转换成 H-指标，然后从最大值、最小值、均值、标准差四个方面进行了比较，结果如表 1 所示。通过表 1 也可以看出 IMOEA/D-ACO 算法在各测试问题上 H-指标的最大值、最小值和均值都是最小的，最小值也是最好的。在标准差方面，虽然只在 kroAB100 这个问题上最小，但在其他几个问题上获得的标准差与最小值相差很小。这说明在这些问题上 IMOEA/D-ACO 算法不但求解质量最佳，稳定性也相对较好。图 3 给出了各算法独立运行 30 次得到的近似 Pareto Front，可以看出算法 IMOEA/D-ACO 在各测试用例上获得的解集明显优于其他算法并且分布比较均匀，更加接近 PF。

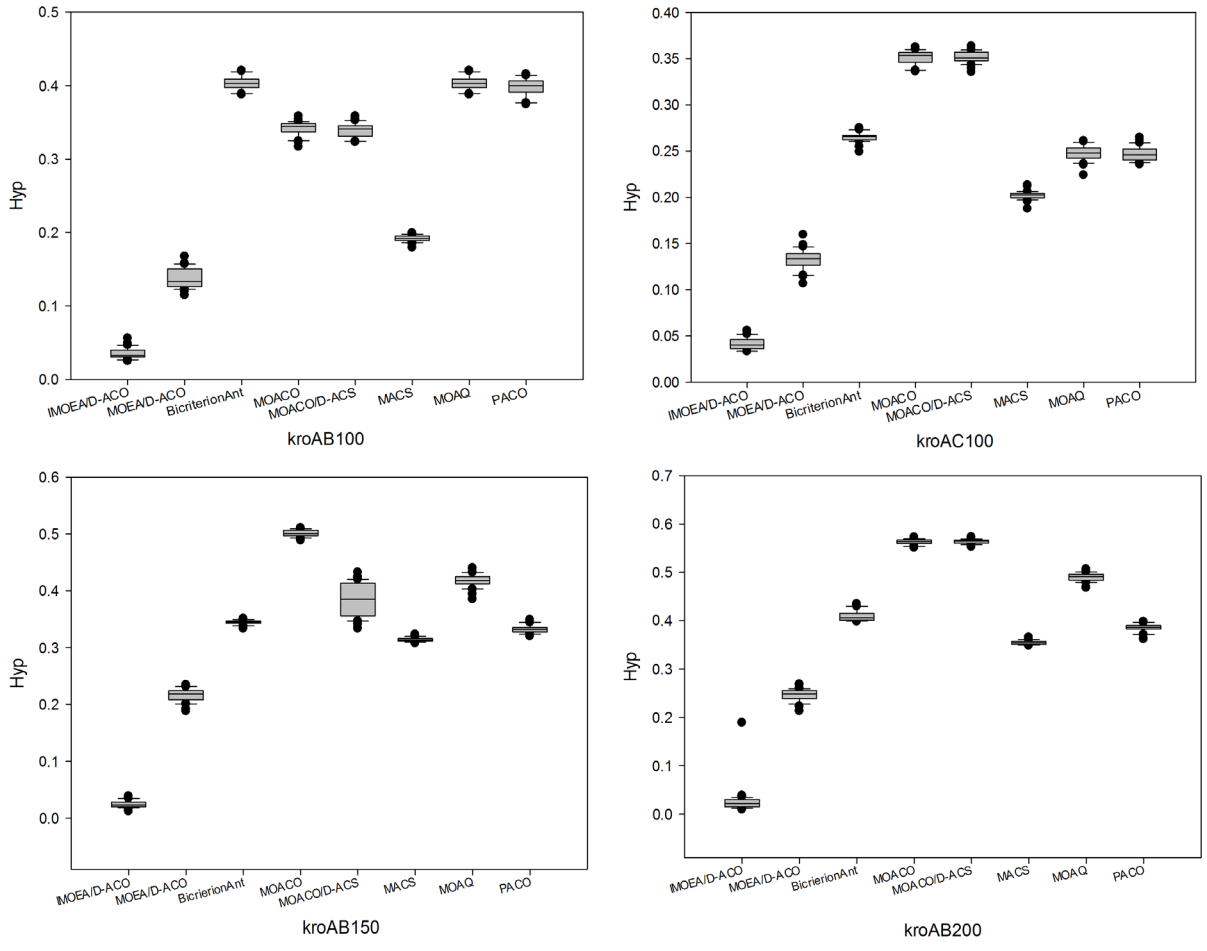


Figure 2. The box plot for H-indicator
图 2. 基于 H 指标的盒子图

Table 1. The comparative results of related algorithm
表 1. 相关算法结果比较

Case	Algorithm	Max	Min	Mean	Standard

Continued

	IMOEA/D-ACO	**0.0562**	**0.0249**	**0.0353**	**0.0071089**
	MOEA/D-ACO	0.1402	0.1172	0.1270	0.0137113
	BicriterionAnt	0.4386	0.4031	0.4201	0.0088096
kroAB100	MACS	0.2230	0.1998	0.2128	0.0052364
	MOACO/D-ACS	0.3675	0.3266	0.3546	0.0089364
	MOACO	0.3678	0.3313	0.3502	0.0101488
	MOAQ	0.4331	0.3958	0.4160	0.0127279
	PACO	0.4309	0.3951	0.4154	0.0106770
	IMOEA/D-ACO	**0.0559**	**0.0332**	**0.0416**	0.0066093
	MOEA/D-ACO	0.1672	0.1049	0.1399	0.0114455
	BicriterionAnt	0.2742	0.2491	0.2645	0.0052990
kroAC100	MACS	0.2102	0.1830	0.1984	**0.0050550**
	MOACO/D-ACS	0.3589	0.3287	0.3465	0.0077633
	MOACO	0.3586	0.3324	0.3467	0.0062761
	MOAQ	0.2559	0.2371	0.2519	0.0078809
	PACO	0.2596	0.2374	0.2506	0.0084581
	IMOEA/D-ACO	**0.0383**	**0.0122**	**0.0243**	0.0067922
	MOEA/D-ACO	0.2661	0.1597	0.1829	0.1555635
	BicriterionAnt	0.3838	0.3065	0.3172	0.0039711
kroAB150	MACS	0.2959	0.2758	0.2868	**0.0036208**
	MOACO/D-ACS	0.4149	0.3129	0.3555	0.0309838
	MOACO	0.5062	0.4666	0.4877	0.0060745
	MOAQ	0.4539	0.4085	0.4309	0.0114018
	PACO	0.3221	0.2932	0.3054	0.0071204
	IMOEA/D-ACO	**0.1890**	**0.0095**	**0.0256**	0.0249480
	MOEA/D-ACO	0.2873	0.2051	0.2585	0.0309102
	BicriterionAnt	0.4521	0.4046	0.4174	0.0100000
kroAB200	MACS	0.3951	0.3556	0.3774	0.0057835
	MOACO/D-ACS	0.5732	0.5532	0.5642	**0.0042448**
	MOACO	0.5765	0.5555	0.4569	0.0044351
	MOAQ	0.5106	0.4933	0.5098	0.0097362
	PACO	0.4233	0.3759	0.3997	0.0095473

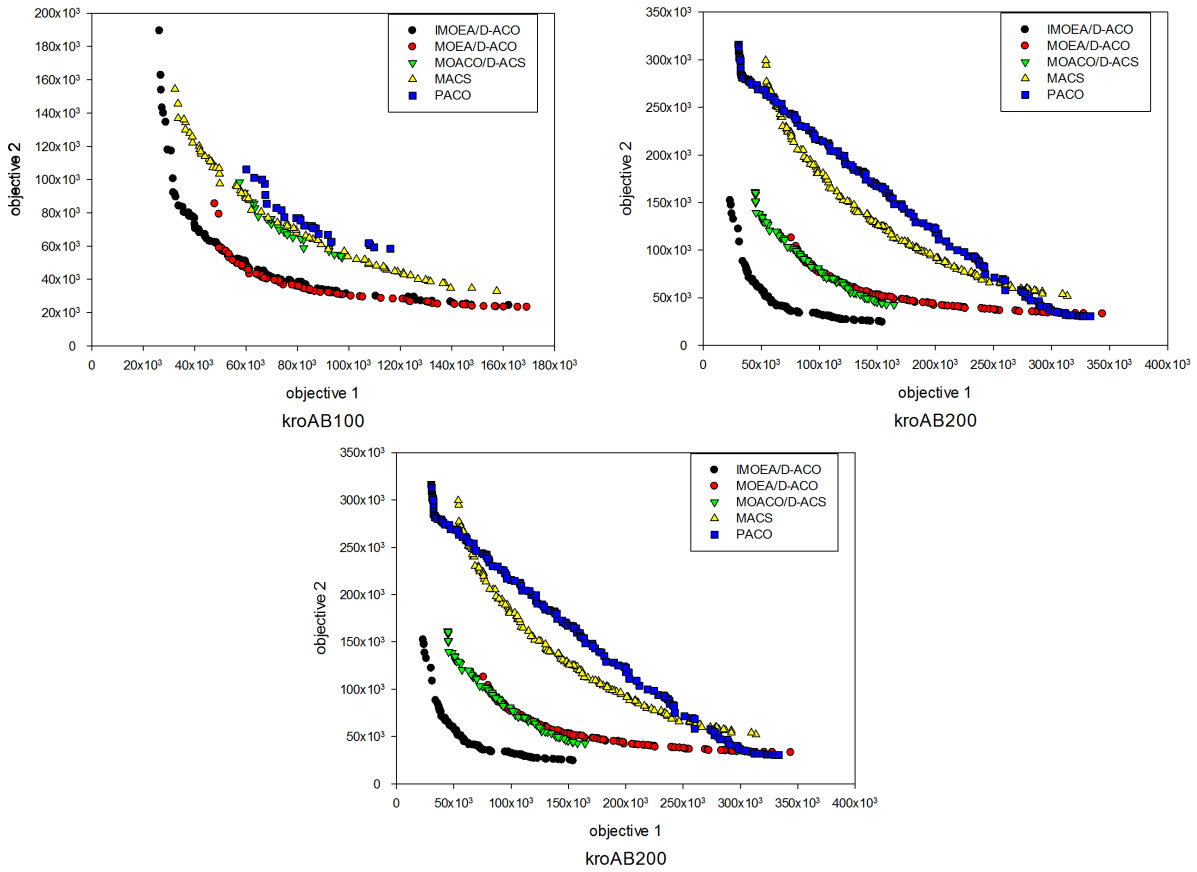


Figure 3. The approximated PFs
图 3. 近似帕累托前沿

4. 总结

已有的基于分解的多目标蚁群优化算法在初始化时采用切比雪夫方法生成子问题并且在整个优化过程中固定不变。基于以上情况，本文在 MOEA/D-ACO 算法的基础上提出了子问题动态削减方法使得在资源有限的前提下，每次迭代时能够根据每个子问题的好坏自动地为每只蚂蚁分配子问题。随着迭代次数的增多，逐步地将不好的子问题删除，将更多的资源用在产生非支配解比较多的子问题上，以达到节约资源的目的。实验结果表明，本文提出的方法是有效的，在搜索资源总量一定的情况下，通过提高搜索资源的利用率能够进一步提高算法的性能。

基金项目

辽宁省博士启动基金项目(2020-BS-152)，沈阳理工大学校级大学生创新训练计划项目(202010144091)。

参考文献

[1] Zuo, L.Y., Shu, L., *et al.* (2015) A Multi-Objective Optimization Scheduling Method Based on the Ant Colony Algorithm in Cloud Computing, *IEEE Access*, **3**, 2687-2699. https://doi.org/10.1109/ACCESS.2015.2508940

[2] Juang, C.-F., Jeng, T.-L. and Chang, Y.-C. (2015) An Interpretable Fuzzy System Learned through Online Rule Generation and Multi-Objective ACO with a Mobile Robot Control Application. *IEEE Transactions on Cybernetics*, **46**, 2706-2718. https://doi.org/10.1109/TCYB.2015.2486779

[3] Wang, L.J. and Shen, J. (2016) Multi-Phase Ant Colony System for Multi-Party Data-Intensive Service Provision. *IEEE Transactions on Services Computing*, **9**, 264-276. https://doi.org/10.1109/TSC.2014.2358213

[4] García-Martínez, C., Cordón, O. and Herrera, F. (2007) A Taxonomy and an Empirical Analysis of Multiple Objective Ant Colony Optimization Algorithms for the Bi-Criteria TSP. *European Journal of Operational Research*, **180**, 116-148. https://doi.org/10.1016/j.ejor.2006.03.041

[5] Barán, B. and Schaerer, M. (2003) A Multiobjective Ant Colony System for Vehicle Routing Problem with Time Windows. *The* 21*st IASTED International Multi-Conference on Applied Informatics*, Innsbruck, 10-13 February 2003, 97-102.

[6] Doerner, K., Gutjahr, W.J., Hartl, R.F., *et al.* (2004) Pareto Ant Colony Optimization: A Metaheuristic Approach to Multiobjective Portfolio Selection. *Annals of Operations Research*, **131**, 79-99. https://doi.org/10.1023/B:ANOR.0000039513.99038.c6

[7] Iredi, S., Merkle, D. and Middendorf, M. (2001) Bi-Criterion Optimization with Multi Colony Ant Algorithms. *International Conference on Evolutionary Multi-Criterion Optimization*, **8**, 359-372. https://doi.org/10.1007/3-540-44719-9_25

[8] López-Ibánez, M. and Stützle, T. (2012) The Automatic Design of Multiobjective Ant Colony Optimization Algorithms. *IEEE Transactions on Evolutionary Computation*, **16**, 861-875. https://doi.org/10.1109/TEVC.2011.2182651

[9] Idid, A. and Tgi, F. (2015) Performance Analysis of the Multi-Objective Ant Colony Optimization Algorithms for the Traveling Salesman Problem. *Swarm and Evolutionary Computation*, **23**, 11-26. https://doi.org/10.1016/j.swevo.2015.02.003

[10] Ke, L., Zhang, Q.F. and Battiti, R. (2013) MOEA/D-ACO: A Multi-Objective Evolutionary Algorithm Using Decomposition and Ant Colony. *IEEE Transactions on Cybernetics*, **43**, 1845-1859. https://doi.org/10.1109/TSMCB.2012.2231860

[11] Ning, J.X., Zhang, Q., Zhang, C.S. and Zhang, B. (2018) A Best-Path-Updating Information-Guided Ant Colony Optimization Algorithm. *Information Science*, **433-434**, 142-162. https://doi.org/10.1016/j.ins.2017.12.047