Ant Based Hyper Heuristics with Space Reduction: A Case Study of the p-Median Problem

Zhilei Ren[1], He Jiang[2,3], Jifeng Xuan[1], and Zhongxuan Luo[1]

[1] School of Mathematical Sciences, Dalian University of Technology
{ren,xuan}@mail.dlut.edu.cn, zxluo@dlut.edu.cn
[2] School of Software, Dalian University of Technology
jianghe@dlut.edu.cn
[3] Department of Computer Science, University of Vermont

Abstract. Recent years have witnessed great success of ant based hyper heuristics applying to real world applications. Ant based hyper heuristics intend to explore the heuristic space by traversing the fully connected graph induced by low level heuristics (LLHs). However, existing ant based models treat LLH in an equivalent way, which may lead to imbalance between the intensification and the diversification of the search procedure. Following the definition of meta heuristics, we propose an Ant based Hyper heuristic with SpAce Reduction (AHSAR) to adapt the search over the heuristic space. AHSAR reduces the heuristic space by replacing the fully connected graph with a bipartite graph, which is induced by the Cartesian product of two LLH subsets. With the space reduction, AHSAR enforces consecutive execution of intensification and diversification LLHs. We apply AHSAR to the p-median problem, and experimental results demonstrate that our algorithm outperforms meta heuristics from which LLHs are extracted.

Keywords: Hyper Heuristics, p-Median, Ant Colony Optimization, Meta Heuristics, Heuristic Space Reduction.

1 Introduction

By definition, a hyper heuristic is the process of using heuristics to choose heuristics to solve the problem in hand [1]. Since its emergence, hyper heuristics have been applied to many problems, such as the timetabling problem [2] and the bin packing problem [3]. The main motivation of hyper heuristics is to conduct search over the heuristic space, rather than directly over the solution space. Among various hyper heuristics, ant based algorithms have attracted much attention in that the pheromone trail provides an intuitive but general representation of the heuristic space. Ant based hyper heuristics explore the heuristic space by traversing the fully connected graph induced by LLHs. However, existing ant based hyper heuristics treat LLHs in an equivalent way, such that LLHs of similar functionalities may be executed consecutively, which may lead to imbalance between the intensification and the diversification of the search procedure.

R. Schaefer et al. (Eds.): PPSN XI, Part I, LNCS 6238, pp. 546–555, 2010.

In this paper, we propose an Ant based Hyper heuristic with SpAce Reduction (AHSAR). Unlike existing ant based hyper heuristics, AHSAR reduces the heuristic space into a subspace by restricting LLHs to be selected from the Cartesian product of two subsets of LLHs. The idea is inspired by the definition of meta heuristics, which interprets the process of meta heuristics as the combination of intensification and diversification mechanisms. With the space reduction, AHSAR is able to adapt the intensification and the diversification of the search procedure effectively. As a case study, AHSAR is applied to a classic NP-hard problem, the p-median problem. Experimental results show that our new algorithm outperforms meta heuristics from which LLHs are extracted.

The paper is organized as follows. In Section 2 we introduce related work of both ant based hyper heuristics and meta heuristics. In Section 3, AHSAR is developed to express iterative intensification and diversification searching methodologies. In Section 4, we apply the new algorithm to the p-median problem, and present the experimental results. Finally, conclusion is given in Section 5.

2 Related Work

2.1 Ant Based Hyper Heuristics

In this subsection, we briefly summarize existing work related to ant based hyper heuristics. In 2005, Burke et al. [4] propose an ant based hyper heuristic to solve the project presentation problem. Alberto et al. [5] apply an ant based hyper heuristic with multiple pheromone matrices for the 2D bin packing problem in the same year. A recent ant based hyper heuristic is proposed by Chen et al. [6] to solve the travelling tournament problem in 2007. Various applications have shown the generality of ant based hyper heuristics. In existing ant based hyper heuristics [4,6], a fully connected graph is firstly constructed, where each vertex represents an LLH, and arcs between vertices indicate invokation sequence relationship between heuristics. Each ant is represented as a sequence of LLHs, which is associated with a solution. At each iteration, artificial ants are constructed by traversing the graph. During the construction phase, the selection of LLHs is guided by the pheromone trail, with each entry representing probability of transition between LLHs. After each LLH sequence is constructed and applied over the corresponding solution, the pheromone information is then updated according to the quality of the solutions obtained.

2.2 Meta Heuristic

Over the last few decades, great efforts have been focused on various meta heuristics, including Genetic Algorithm (GA) [7], ACO [8], Tabu Search (TS) [9, 10], Variable Neighborhood Search (VNS) [11], Greedy Randomized Adaptive Search (GRASP) [12], etc. Despite appearing to be far different from each other, these algorithms share a few common aspects [13]. By concept, a meta heuristic algorithm is defined as "an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and

exploiting the search space" [14], or the combination of intensification and diversification. In the definition, exploitation (also called intensification) indicates mechanisms that conduct intensive search in order to improve the quality of solutions; while exploration (also called diversification) refers to various mechanisms that lead the search to diverse regions of the search space. Some ways of achieving intensification and diversification are listed in Table 1.

Table 1. Some ways to achieve intensification and diversification in meta heuristics

Meta Heuristics	Intensification	Diversification
GA	Survival selection	Crossover and mutation
ACO	Local search	Initialization with pheromone
	Pheromone accumulation	Pheromone evaporation
VNS	Local search	Shake
GRASP	Local search	Greedy randomized initialization

As presented in Table 1, meta heuristic algorithms intend to balance the intensification and the diversification mechanisms by combining those intensification and diversification LLH operators. During the intensification process, heuristics such as local search operators are usually applied so as to improve the quality of solutions; while during the diversification process, various perturbing heuristics such as crossover, mutation and shake are employed to restart the search procedure in new regions of the search space.

3 Ant Based Hyper Heuristic with Space Reduction

In this section, we discuss strategies of exploring the heuristic space, and propose our new algorithm, AHSAR. For ant based models, how to traverse the graph derived by LLHs so as to explore the heuristic space plays an essential role. Burke et al. [4] discuss two traversing criteria: the Any Moves (AM) hyper heuristics that accept any LLH sequences, and the Only Improving (OI) hyper heuristics that only accept LLH sequences that improve solution quality. They claim that AM outperforms OI, in that OI strategy provides no diversification mechanism [4]. However, AM strategy treats all LLHs in an equivalent way, thus heuristics with similar functionalities may be invoked consecutively. This may diminish the effect of some LLHs. For example, if a random restart heuristic is invoked immediately after a greedy heuristic, the execution of this greedy heuristic is generally a waste of time. Based on this observation, we propose our new ant model AHSAR, which directly follows the definition of meta heuristics. To enforce the consecutive execution of intensification and diversification LLHs, we replace the fully connected graph induced by the whole LLH set with a bipartite graph derived by two subsets of LLHs.

Without loss of generality, given a minimization problem, let S be the solution space, with objective function $f : S \mapsto \mathbb{R}$, and a heuristic is defined as a function

$h : S \mapsto S$. Let H be the set of LLHs, and $I = \{i | i \in H, \forall s \in S, f(i(s)) \leq f(s)\}$ and $D = \{d | d \in H, \exists s \in S, f(d(s)) > f(s)\}$ be the subset of H that provide intensification and diversification mechanisms, respectively. For heuristics with more than one input solution, such as the crossover of GA, we only need to replace the objective function f with some other evaluation function. Instead of traversing the fully connected graph in search for LLHs, at each iteration of AHSAR, each ant selects a tuple $\langle i, j \rangle$ from the Cartesian product $I \times D$, and then the chosen heuristics are applied. In order to guide the LLH selection, a pheromone matrix τ is incorporated, with each entry τ_{ij} measuring the desirability of selecting $\langle i, j \rangle$. The higher the pheromone value is, the higher the probability of choosing the corresponding tuple will be. In contrast to existing ant based hyper heuristics, the scale of pheromone matrix decreases significantly from $|H \times H|$ to $|I \times D|$.

In addition to the pheromone matrix τ, we also incorporate the heuristic information to balance the probability of choosing each LLH. The idea is intuitive, which is used in other ant based hyper heuristics [4] as well: the computational complexity of LLHs may be quite different from each other, so that we should introduce some mechanism to penalize those time consuming LLHs. For an extreme example, if we consider the exhaustive search as an operator, it can certainly obtain the best solution quality. However, the running time may increase exponentially as the scale of the problem instance grows. Although LLHs with high complexity may achieve high quality solutions, we should also take running time into account. In AHSAR, η_{ij} is defined as:

$$\eta_{ij} = \frac{T_{norm}}{T_i + T_j} \tag{1}$$

where T_i and T_j indicate the running time of LLH i and j, respectively, and T_{norm} is a normalizer in order to balance those problem instances of different scale. In practice, it can be set with the running time of an LLH, which can be determined during the initialization of the algorithm. The definition of η_{ij} implies that we prefer those LLHs with low complexity, with respect to the efficiency.

With τ and η, we can define the probability of choosing the tuple $\langle i, j \rangle$ at each iteration:

$$P_{ij} = \frac{\eta_{ij} \cdot \tau_{ij}}{\sum_{i \in I} \sum_{j \in D} \eta_{ij} \cdot \tau_{ij}} \tag{2}$$

At each iteration, after the tuple $\langle i, j \rangle$ is chosen and applied on the solution associated with each ant k, the pheromone τ is updated using the following rule:

$$\tau_{ij} = \begin{cases} \rho \cdot \tau_{ij} + \dfrac{C^{best}}{C^k} & \text{ant } k \text{ chooses } \langle i, j \rangle, \\ \rho \cdot \tau_{ij} & \text{otherwise.} \end{cases} \tag{3}$$

where C^k and C^{best} represent the objective function of the solution corresponding with ant k and the current best solution, respectively, and ρ indicates the evaporation rate.

```
Algorithm AHSAR
Input: maximum iteration N,
       number of ant K,
       number of elite ant k*,
       evaporation rate ρ
Output: solution s
Begin
(1) Initialize τ and η with sufficiently small random values
(2) for iteration = 1 to N do
  (3) for k = 1 to K do
    (3.1) Choose ⟨i, j⟩ with probability defined in equation (2)
    (3.2) Apply LLH tuple ⟨i, j⟩ to the solution
          associated with ant k
    (3.3) Update η with equation (1)
    (3.4) if a better solution has been obtained
    (3.4.1) Record the current best solution with s and update C^best
  (4) Select k* solutions with highest quality
  (5) for k = 1 to k* do
    (5.1) Update pheromone with equation (3)
(6) return the best solution obtained s
End
```

Fig. 1. Pseudo Code of AHSAR

Fig. 1 presents the pseudo code of AHSAR. First of all, initialization heuristics are treated as special cases of diversification heuristics in order to make the algorithm compact and easier to implement. Thus at each iteration, after the tuple $\langle i, j\rangle$ is selected according to equation (2), the diversification operator j is applied before i. Second, crossover is a pairwise heuristic that requires two parent solutions, thus if the crossover operator is chosen, an extra solution is randomly selected from the population. Third, in step (4), k^* out of K ants are selected, and only these ants are allowed to release pheromone. This mechanism is similar to the survival selection of GA and the elite ant strategy in ACO. We introduce this strategy to penalize those ineffective heuristics, in that the pheromone evaporation mechanism will decrease the probability of choosing them. Finally, an interesting byproduct of the algorithm in this study is that AHSAR actually provides a unified framework under which many existing meta heuristics can be considered as special cases. Although the definition of the two subsets of LLH may not be quite precise (for example, most intensification LLHs are local search operators), the proposed model can express a wide range of meta heuristics. For example, by fixing the value of the pheromone entry corresponding to $\langle$Local search, Greedy randomized initialization$\rangle$ to be 1, and that of all the other entries to be 0, AHSAR will degenerate into GRASP. Specifically, if we are considering individual based algorithms such as VNS, we just set parameter $k = 1$, which makes the ant model similar to Fast Ant (FANT) [15].

4 Experimental Results: A Case Study of the p-Median Problem

In this section, we apply our AHSAR to the p-median problem. The reasons we choose the p-median problem are as follows. Firstly, it is a classic NP-hard problem from location theory with wide applications ranging from industry to data mining. Since location theory is a new domain for hyper heuristics, the proposed algorithm demonstrates the generality and the extensibility of hyper heuristics. Secondly, for the p-median problem, there exists many meta heuristics from which LLHs can be extracted, such as VNS [16], ACO [17], and GA [18].

Given a set L of m facilities, a set U of n users, and an $n \times m$ matrix C with the cost traveled c_{ij} for satisfying the demand of the user located at i from the facility located at j, for all $j \in L$ and $i \in U$. The objective of the p-median problem is to minimize the sum of these costs:

$$\min \sum_{i \in U} \min_{j \in J} c_{ij} \tag{4}$$

All LLHs are extracted from those existing meta heuristics mentioned above. Intensification heuristics are listed as follows.

- Interchange: Interchange is first proposed in [19], and widely used in meta heuristics, such as VNS [16] and GRASP [20]. The heuristic iteratively swaps facilities aiming to reduce objective function, until no move can be applied.
- LK(2): LK(k) is extracted from ACO algorithm [17], in which k is a depth parameter. Traversing an LK(k) neighborhood involves k swaps, which is k times that of interchange. For LK(2), parameter k is set to be 2.
- LK($m/2$): Same as LK(k), with $k = m/2$, where m indicates the number of facilities.
- LK(m): Same as LK(k), with $k = m$.

Besides intensification heuristics, diversification heuristics are also listed.

- Crossover and mutation: These two heuristics are extracted from GA [18]. Note that crossover requires two input solutions.
- Initialization with pheromone (AntInit): At each iteration of ACO [17], each ant is constructed with probability according to the pheromone trail. In this study, there are two pheromone matrices, one for the solution space, while the other for the heuristic space.
- Shake: Shake is proposed in VNS [16], and can be viewed as a special case of mutation, whose input is provided by the current best solution.
- Random: Random is actually an initialization operator, which can provide diversification functionality as well.
- Greedy: Greedy is also an initialization operator. Greedy starts with an empty solution, followed by solving p 1-median problems.
- Random plus greedy (RPG): As mentioned above, greedy is a deterministic heuristic, thus in [20], randomness is combined with greedy operator.

Table 2. Results of Algorithms on ORLIB instances

id	ACO	time	GA	time	MStart	time	RANDH	time	AHFAM	time	AHSAR	time
1	5819	2.75	5819	0.81	5819	0.25	5819	2.79	5819	0.77	5819	0.72
2	4093	1.68	4102	0.86	4093	0.26	4093	1.66	4093	0.57	4093	0.52
3	4250	1.70	4250	0.83	4250	0.25	4250	1.70	4250	0.53	4250	0.46
4	3034	1.08	3038	0.83	3034	0.44	3034	1.08	3034	0.40	3034	0.45
5	1355	0.92	1362	0.87	1355	0.36	1355	0.91	1355	0.36	1355	0.37
6	7824	17.84	7824	0.81	7824	0.81	7824	17.83	7824	5.21	7824	4.26
7	5631	8.13	5631	0.83	5631	0.69	5631	8.15	5631	2.07	5631	2.09
8	4445	5.75	4445	0.82	4445	0.84	4445	5.87	4445	1.36	4445	1.25
9	2734	3.33	2734	0.86	2734	1.23	2734	3.20	2734	1.06	2734	1.07
10	1255	3.12	1272	0.86	1255	1.06	1255	3.16	1255	1.06	1255	1.10
11	7696	40.33	7696	0.82	7696	1.48	7696	39.85	7696	9.79	7696	5.98
12	6634	24.21	6634	0.84	6634	1.33	6634	24.20	6634	6.71	6634	5.80
13	4374	7.43	4381	0.85	4374	2.21	4374	7.45	4374	2.20	4374	2.30
14	2968	7.15	2975	0.89	2968	2.37	2968	6.85	2968	2.00	2968	1.87
15	1729	7.28	1736	0.88	1729	2.16	1729	7.22	1729	2.14	1729	2.27
16	8162	121.40	8162	0.83	8162	2.85	8162	121.66	8162	30.71	8162	16.78
17	6999	60.34	6999	0.84	6999	2.04	6999	61.10	6999	15.48	6999	7.58
18	4809	12.70	4815	0.87	4809	4.51	4809	12.78	4809	3.61	4809	3.76
19	2845	10.34	2853	0.94	2846	3.52	2845	10.30	2845	3.17	2845	3.09
20	1789	11.93	1795	0.94	1789	3.53	1789	11.87	1789	3.51	1789	3.22
21	9138	178.31	9138	0.82	9138	4.04	9138	178.78	9138	50.67	9138	28.92
22	8579	137.75	8579	0.94	8579	3.50	8579	139.86	8579	32.11	8579	16.47
23	4619	17.37	4627	0.95	4619	8.13	4619	17.15	4619	5.58	4619	5.78
24	2961	16.04	2977	0.90	2961	5.60	2961	16.36	2961	4.94	2961	4.46
25	1828	21.51	1847	0.97	1828	5.87	1828	20.77	1828	6.24	1828	6.01
26	9917	393.83	9924	1.10	9917	7.63	9917	395.34	9917	87.15	9917	46.96
27	8307	270.92	8307	0.94	8307	5.64	8307	277.16	8307	65.12	8307	28.12
28	4498	24.65	4520	1.06	4498	13.09	4498	24.66	4498	7.73	4498	7.80
29	3033	26.05	3042	1.18	3033	8.57	3033	25.08	3033	7.47	3033	7.29
30	1989	36.01	2004	1.21	1994	9.29	1989	34.70	1989	9.85	1989	7.55
31	10086	666.73	10086	1.00	10086	12.17	10086	663.73	10086	154.79	10086	108.91
32	9297	420.62	9297	0.91	9297	8.57	9297	420.05	9297	103.52	9297	34.86
33	4700	34.36	4723	1.04	4700	20.84	4700	33.76	4700	11.30	4700	11.82
34	3013	42.16	3032	1.30	3014	12.58	3013	40.11	3013	10.78	3013	10.69
35	10400	1124.54	10400	1.29	10400	17.44	10400	1119.59	10400	256.02	10400	127.66
36	9934	815.42	9951	1.30	9934	12.28	9934	804.58	9934	181.60	9934	98.18
37	5057	49.25	5071	1.24	5057	31.29	5057	48.79	5057	15.97	5057	18.26
38	11060	2065.51	11105	0.87	11060	27.03	11060	2106.39	11060	396.10	11060	206.86
39	9423	1027.39	9423	0.87	9423	15.81	9423	1027.65	9423	236.53	9423	106.17
40	5128	63.13	5134	1.09	5129	42.63	5128	64.00	5128	20.00	5128	23.21

All experiments of this paper are performed on a Pentium IV 3.2 GHz PC with 4GB memory, running GNU/Linux with kernel 2.6.32. All the codes are implemented in C++, compiled using gcc 4.4.3 with flag -O2. Running time is measured in seconds, calculated using clock() function. For pseudo random number we use rand() from standard library. The benchmark instance set consists of 40 instances from ORLIB [21], and 10 instances from TSPLIB [22] (We consider instance f1400, with various p indicated by *id* in Table 3). To evaluate the effectiveness and the efficiency of our algorithm, we implement several algorithms for comparison, including ACO [17], GA [18], a multistart local search (denoted as MStart), and a basic hyper heuristic algorithm that randomly chooses LLHs at each iteration (denoted as RANDH). For ant based hyper heuristics, besides AHSAR, we also implement the version that traverses over the fully connected graph with AM strategy (denoted as AHFAM), as discussed in [4].

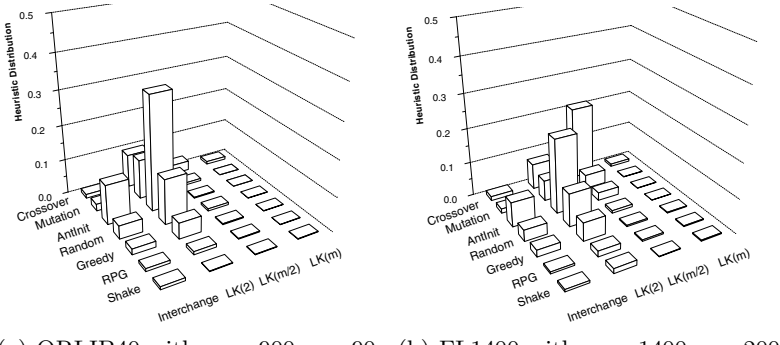
We first compare performance between existing meta heuristics and hyper heuristics proposed in this paper. For all algorithms, parameters are set with same values (population of solutions $K = 10$, maximum iteration $N = 100$, number of elite solutions $k^* = 5$ for ant based hyper heuristics and GA, and evaporation rate $\rho = 0.1$) in order to compare both effectiveness and efficency of each algorithm. For the results presented in Table 2 and Table 3, solutions with best qualities are underscored for each instance. We can observe that solution quality of AHSAR outperforms meta heuristic algorithms from which LLH

Table 3. Results of Algorithms on TSPLIB instances

id	ACO	time	GA	time	MStart	time	RANDH	time	AHFAM	time	AHSAR	time
50	29090.22	272.79	29328.30	1.31	29090.23	46.73	29090.23	273.47	29090.22	72.35	29090.22	100.88
100	16559.32	143.30	16759.88	1.59	16569.18	93.57	16562.20	145.16	16566.57	51.46	16552.22	72.74
150	12045.58	178.18	12100.59	1.68	12054.59	45.88	12049.74	175.09	12051.25	46.01	12026.43	77.38
200	9361.34	176.85	9409.70	2.34	9385.97	46.82	9365.61	178.43	9362.59	47.74	9359.05	64.23
250	7747.84	206.06	7781.28	2.51	7761.73	50.49	7748.78	209.76	7743.91	40.05	7742.67	62.97
300	6629.41	224.28	6674.16	2.08	6649.03	52.03	6629.98	227.09	6627.53	44.80	6623.81	107.64
350	5743.23	279.56	5784.00	3.13	5766.73	55.75	5738.79	280.16	5736.31	50.02	5723.47	93.28
400	5030.44	326.14	5093.38	2.72	5058.47	60.09	5026.96	324.66	5017.05	72.92	5009.67	116.64
450	4478.42	440.36	4525.08	3.06	4500.72	63.81	4480.79	452.71	4481.81	88.14	4476.72	112.18
500	4052.17	489.50	4086.64	3.59	4068.32	65.72	4053.38	495.88	4053.94	92.40	4048.96	194.46

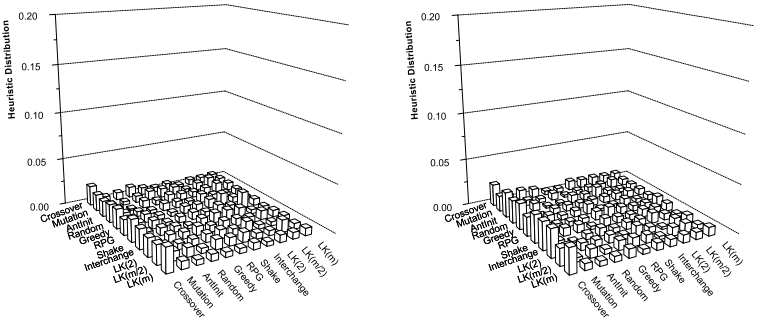
are extracted. AHSAR obtains best solution for all benchmark instances. When running time is concerned, we can see that both AHSAR and AHFAM run faster than ACO. Although ACO obtains best quality among meta heuristics presented, high complexity of LK(m) that ACO employs makes it quite time consuming. With help of the heuristic information η in our algorithm, running time is significantly reduced. We can also observe the effecacy of the pheromone matrix, in that both AHSAR and AHFAM achieve solutions with higher quality than RANDH in less computation time. Besides, the comparison between AHFAM and AHSAR implies the effect of the space reduction mechanism. With the reduction mechanism, the combination of intensification and diversification contributes to the better quality of the solution obtained. Although AHSAR runs slower than AHFAM for some instances, the quality-time trade-off is worthy.

More insights can be gained by analyzing the distribution of LLHs over different instances. Fig. 2 depicts probabilities of selecting different combinations of LLHs during one run of AHSAR. From Fig. 2, we can see that given different problem instances, the distribution of LLH combinations exhibits different patterns. For example, over orlib40 and fl1400 with $p = 200$, the probability of selecting $\langle \text{LK}(2), \text{AntInit} \rangle$ varies from 40% to 22%. However, LLH distribution of AHFAM doesn't exhibit different pattern over different instances, as is illustrated in Fig. 3. The observation implies that our new algorithm is self-adaptive. Besides, Fig. 2 implies the effect of the heuristic matrix η and the elite ant



(a) ORLIB40 with $m = 900, p = 90$ (b) FL1400 with $m = 1400, p = 200$

Fig. 2. ARSAR's distribution of LLHs over different instances



(a) ORLIB40 with $m = 900, p = 90$ (b) FL1400 with $m = 1400, p = 200$

Fig. 3. AHFAM's distribution of LLHs over different instances

strategy. From Fig. 2 we can see that the LK(m) operator is seldom chosen, in that its complexity is too high; while the probability of selecting LLHs such as Random and RPG is also low, due to their poor performance.

5 Conclusion

In this paper a new ant based hyper heuristic with space reduction is developed. The main contribution of this paper can be concluded as follows. (1) A new ant model is proposed. We explicitly follow the definition of meta heuristics by dividing LLHs according to their functionalities, and the search space of LLHs is reduced by replacing the fully connected graph constructed by all LLHs with a bipartite graph derived by two subsets of LLHs, which can be explored more effectively. (2) We apply the proposed algorithm to the p-median problem for the first time, which demonstrate the generality of hyper heuristics. (3) Numerical results show that our algorithm outperforms the meta heuristics from which the new algorithm is derived.

Acknowledgments. Our work is partially supported by the Natural Science Foundation of China under Grant No. 60805024, the National Research Foundation for the Doctoral Program of Higher Education of China under Grant No. 20070141020.

References

1. Burke, E.K., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E., Woodward, J.R.: A Classification of Hyper-heuristics Approaches. In: Gendreau, M., Potvin, J.Y. (eds.) Handbook of Metaheuristics, International Series in Operations Research & Management Science. Springer, Heidelberg (2009) (in press)
2. Burke, E.K., McCollum, B., Meisels, A., Petrovic, S., Qu, R.: A Graph-based Hyper-heuristic for Educational Timetabling Problems. European Journal of Operational Research 176(1), 177–192 (2007)

3. Privosnik, M.: The Scalability of Evolved On Line Bin Packing Heuristics. In: IEEE Congress on Evolutionary Computation, CEC 2007, pp. 2530–2537 (2007)
4. Burke, E.K., Kendall, G., Silva, D.L., O'Brien, R., Soubeiga, E.: An Ant Algorithm Hyperheuristic for the Project Presentation Scheduling Problem. In: The 2005 IEEE Congress on Evolutionary Computation, vol. 3, pp. 2263–2270 (2005)
5. Cuesta-Cañada, A., Garrido, L., Terashima-Marín, H.: Building Hyper-heuristics Through Ant Colony Optimization for the 2D Bin Packing Problem. In: Knowledge-Based Intelligent Information and Engineering Systems, pp. 654–660. Springer, Heidelberg (2005)
6. Chen, P.C., Kendall, G., Berghe, G.: An Ant Based Hyper-heuristic for the Travelling Tournament Problem. In: IEEE Symposium on Computational Intelligence in Scheduling, SCIS 2007, pp. 19–26 (2007)
7. Holland, J.H.: Adaptation in Natural and Artificial Systems. MIT Press, Cambridge (1992)
8. Dorigo, M., Maniezzo, V., Colorni, A.: Ant System: Optimization by a Colony of Cooperating Agents. IEEE Transactions on Systems, Man, and Cybernetics, Part B 26(1), 29–41 (1996)
9. Glover, F.: Tabu search – Part I. ORSA J. on Computing 1, 190–206 (1989)
10. Glover, F.: Tabu search – Part II. ORSA J. on Computing 2, 4–32 (1990)
11. Hansen, P., Mladenović, N.: Variable Neighborhood Search: Principles and Applications. European Journal of Operational Research 130(3), 449–467 (2001)
12. Feo, T.A., Resende, M.G.C.: Greedy Randomized Adaptive Search Procedures. Journal of Global Optimization 6(2), 109–133 (1995)
13. Taillard, É.D., Gambardella, L.M., Gendreau, M., Potvin, J.Y.: Adaptive Memory Programming: A Unified View of Metaheuristics. European Journal of Operational Research 135(1), 1–16 (2001)
14. Osman, I.H., Laporte, G.: Metaheuristics: A Bibliography. Annals of Operations Research 63(5), 511–623 (1996)
15. Taillard, É.D.: Ant Systems. In: Pardalos, P., Resende, M.G.C. (eds.) Handbook of Applied Optimization, pp. 130–137. Oxford Univ. Press, Oxford (2002)
16. Hansen, P., Mladenović, N.: Variable Neighborhood Search for the p-Median. Location Science 5(4), 207–226 (1997)
17. Kochetov, Y., Levanova, T., Alekseeva, E., Loresh, M.: Large Neighborhood Local Search for the p-Median Problem. Yugoslav Journal of Operations Research 15(1), 53–63 (2005)
18. Alp, O., Erkut, E., Drezner, Z.: An Efficient Genetic Algorithm for the p-Median Problem. Annals of Operations Research 122, 21–42 (2003)
19. Teitz, M.B., Bart, P.: Heuristic Methods for Estimating the Generalized Vertex Median of a Weighted Graph. Operations Research 16(5), 955–961 (1968)
20. Resende, M.G.C., Werneck, R.F.: A Hybrid Heuristic for the p-Median Problem. Journal of Heuristics 10(1), 59–88 (2004)
21. Beasley, J.E.: A Note on Solving Large p-Median Problems. European Journal of Operational Research 21, 270–273 (1985)
22. Reinelt, G.: TSPLIB–A Traveling Salesman Problem Library. ORSA Journal on Computing 3, 376–384 (1991)