A Sampling based FANT for the 3-Dimensional Assignment Problem

He Jiang, Zhilei Ren, Yan Hu

***Abstract*—In this paper, we proposed a sampling based FANT (S-FANT) for the 3-Dimensional Assignment Problem (AP3). The AP3 is a well-known NP-hard problem, which aims to choose n disjoint triplets with minimum cost from 3 disjoint sets of size n. Due to its intractability, many heuristics have been proposed to obtain near optimal solutions in reasonable time. Since the solution space size of the AP3 is $(n!)^2$, traditional FANT algorithms can't work well for the AP3. In this paper, we showed that, those triplets frequently contained by local optimal solutions are likely to belong to global optimal solutions. Therefore, those triplets can help the ant to converge faster to global optimal solutions. Upon the observation above, the S-FANT consists of two phases. In the sampling phase, a multi-restart scheme is employed to generate local optimal solutions. After that, the pheromone is initialized according to the frequency of triplets appearing in those local optimal solutions. In the FANT phase, a standard FANT algorithm is conducted to explore for better solutions. Extensive experimental results on the standard AP3 benchmark indicated that the new algorithm outperforms the state-of–the-art heuristics in terms of solution quality. Work of this paper not only provides a new efficient heuristic for the AP3, but shows a promising way to design FANT algorithms for those NP-hard problems with large solution space.**

I. Introduction

The 3-Dimensional Assignment Problem (AP3) was first introduced by Pierskalla (1967, 1968). It can be viewed as an optimization problem on a complete tripartite graph $K_{n\times n\times n} = (I \cup J \cup K, (I\times J)\cup(I\times K)\cup(J\times K))$, where I, J and K are disjoint sets of size n. The cost of choosing a triplet (i, j, k) from $I\times J\times K$ is denoted by c_{ijk}. The aim of AP3 is to choose n disjoint triplets (i, j, k) so that the total cost is minimized. The AP3 problem can be formulated as $\min \sum_{i=1}^{n} c_{ip(i)q(i)}$, $p,q \in \pi_N$, where π_N denotes the set of all permutations on the set of integers $N = \{1,2,\ldots,n\}$. Then any pair of permutations (p,q) ($p,q \in \pi_N$) is a feasible solution to an AP3 instance of size n.

Since the AP3 has been proven to be NP-hard, many heuristic algorithms have been proposed to solve it, including Balas and Saltzman (1991), Burkard and Rudolf (1993), Crama and Spieksma (1992), Pardalos and Pitsoulis (2000), Voss(2000), Aiex, Resende, Pardalos and Toraldo (2005). Among these algorithms, GRASP with path relinking proposed by Aiex, Resende, Pardalos and Toraldo (2005) can obtain better results than other existing heuristics.

Since the solution space size of the AP3 is $(n!)^2$, traditional FANT algorithms can't work well for the AP3. However, we showed by experiments that those triplets in global optimal solutions tend to appear in local optimal solutions with great probability. Thus, these triplets can guide the ant to find global optimal solutions in search procedure. Therefore, a sampling based FANT (S-FANT) is proposed for the AP3 in this paper. In contrast to traditional FANT algorithms, our S-FANT consists of two phases. The sampling phase employs a multi-restart scheme to generate local optimal solutions. After that, the pheromone is initialized according to the frequency of triplets appearing in those local optimal solutions. In the FANT phase, a standard FANT algorithm is implemented to search for better solutions. Extensive experimental results indicated that our new algorithm outperforms the state-of–the-art heuristics in terms of solution quality on the standard AP3 benchmark--Balas and Saltzman Dataset.

The rest of this paper is organized as follows. Section II explores the relationship between local optimal solutions and global optimal solutions. In Section III, we shall propose the S-FANT framework. Experimental results are reported in Section IV. Finally, we conclude this paper in Section V.

II. Solution Space Investigation for AP3

In this section, we shall investigate the relationship between local optimal solutions and global optimal solutions.

Boese (1995) observed in the traveling salesman problem (TSP) that a local optimal solution tends to have 80% common edges shared with a global optimal solution. Similar phenomena were also found in the graph partitioning problem (Merz and Freisleben, 2000), and the job shop scheduling problem (Reeves, 1999). Such observations have led to the "big valley" structure, which suggests that extensively many local optimal solutions form clusters around the optimal solutions.

Similar to the work of Boese, we examined the feature of local optimal solutions in AP3 as follows.

Firstly, 1000 local optimal solutions (denoted by $s_1, s_2, \cdots, s_{1000}$) were generated to several typical instances (bs_8_1.dat, bs_10_1.dat and bs_12_1.dat) from Balas and Saltzman Dataset (Balas and Saltzman, 1991), by 1000 runs of a Random Local Search (RLS) algorithm (see Algorithm 1). RLS starts with a randomly generated initial solution (step (1)-step (3)), followed by a Hungarian local search

This work was supported in part by the National Natural Science Foundation of China under Grant 60673046 and 60673066, the Natural Science Foundation of Liaoning Province under Grant 20051082, and the Gifted Young Foundation of Dalian University of Technology.

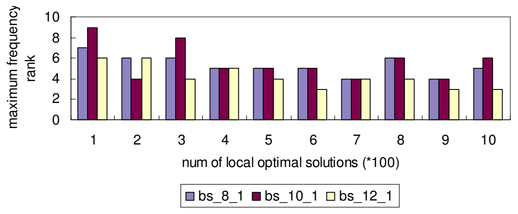
He Jiang is with School of Software, Dalian University of Technology, Dalian, 116621, P.R.China (phone: 86-411-81381830; fax: 86-411-87571567; e-mail: jianghe@dlut.edu.cn).

Zhilei Ren is with School of Software, Dalian University of Technology, Dalian, 116621, P.R.China (e-mail: cos_alpha@hotmail.com).

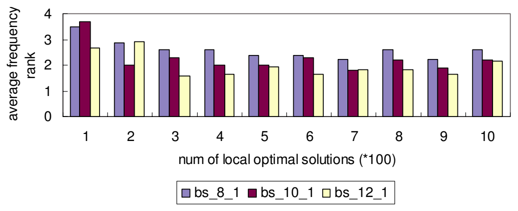
Yan Hu is with School of Software, Dalian University of Technology, Dalian, 116621, P.R.China (e-mail: wohuyan@gmail.com).

(step (4)-step (5)) which was proposed in (Huang and Lim, 2006) by projecting an AP3 instance to a linear assignment problem.

```
Algorithm 1: RLS for AP3
Input: AP3 instance
Output: solution $s'$
Begin
  (1) for $i = 1$ to $n$ do $p[i] = i$, $q[i] = i$;
  (2) for $i = 1$ to $n$ do
        let $j$ be a random integer between 1 and $n$;
        swap $p[i]$ and $p[j]$;
  (3) for $i = 1$ to $n$ do
        let $j$ be a random integer between 1 and $n$;
        swap $q[i]$ and $q[j]$;
  (4) let $s = \{(i, p[i], q[i]) | 1 \le i \le n\}$;
  (5) obtain the local optimal solution $s'$ by applying
  the Hungarian local searcher to $s$;
End
```



(a) Maximum frequency rank

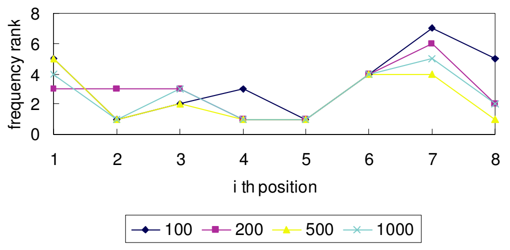


(b) Average frequency rank

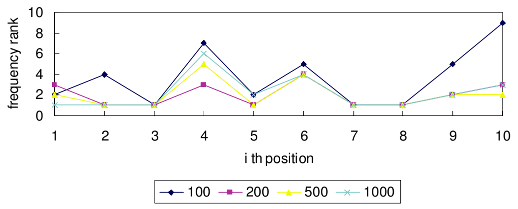
Fig. 1. Frequency rank of triplets in global optimal solutions vs. num of local optimal solutions.

Secondly, we investigated the relationship between the triplets in global optimal solutions and local optimal solutions (see Fig. 1). Let $T(i)$ be the set of triplets containing position $i \in I$ in local optimal solutions. We ranked all the triplets $(i, j, k) \in T(i)$ for every $i \in I$, according to the frequency of (i, j, k) appearing in local optimal solutions. In Fig. 1(a), the maximum frequency rank of all triplets in global solutions is plotted against the num of local optimal solutions. It can be concluded that the maximum frequency ranks of bs_8_1, bs_10_1, and bs_12_1 vary between 3 and 9. In Fig. 1(b), the average frequency rank of all triplets in global solutions is plotted against the num of local optimal solutions. Obviously, the average frequency ranks of bs_8_1, bs_10_1, and bs_12_1 vary between 1 and 4.

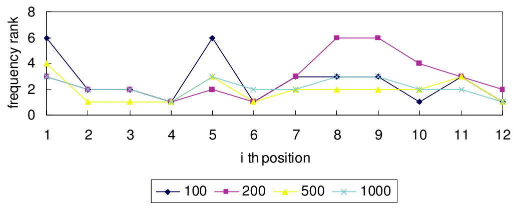
Fig. 2 demonstrates the frequency rank of triplets in global optimal solutions against the num of local optimal solutions for every $i \in I$. In every graph, four curves are plotted respectively for 100, 200, 500 and 1000 local optimal solutions. Fig. 2(a) indicates that the frequency ranks of triplets in global solutions vary between 1 and 7 for instance bs_8_1. Fig. 2(b) shows that the frequency ranks vary between 1 and 9 for instance bs_10_1. And Fig. 2(c) indicates that the frequency ranks vary between 1 and 6.



(a) bs_8_1



(b) bs_10_1



(c) bs_12_1

Fig. 2. Frequency rank of triplets in global optimal solutions for every $i \in I$.

Inspired by the relationship between local optimal solutions and global optimal solutions, we can then guide the ant in FANT by those triplets with high frequencies.

III. S-FANT For AP3

Upon the observation in Section II, we shall present the S-FANT for the AP3 in this section.

The traditional ACO framework was proposed by Dorigo, Maniezzo and Colorni (1991). Its underlying idea was inspired by the behavior of natural ants, which are capable of finding shortest path from colony to food by communicating using indirect information called pheromone.

In contrast to ACO, the key idea of FANT (Taillard and Gambardella, 1997) is to make the ant system as simple as possible, meanwhile incorporating diversification and intensification mechanism. This is realized as follows: On one hand, FANT reinforces the attractiveness of the path of current best solution systematically to search the neighborhood of current best solution; On the other hand, when the current best solution doesn' t improve in a long time, which means that FANT has converged, then pheromone information is reinitialized to reduce the weight of current best solution so as to diversify the search procedure. FANT framework is composed of four components: memory structure, construction of a provisory solution, local search for improving the provisory solution, and the memory update way.

```
Algorithm 2: S-FANT for AP3
Input: AP3 instance, $k$, heuristic H, iterations $M$
Output: solution $s^*$
Begin
  // the sampling phase
  (1) $c^* = +\infty$, $s = \varnothing$
  (2) for $i = 1$ to $k$ do
      (2.1) obtain a solution $s_i$ with H;
      (2.2) if $c(s_i) < c^*$ then $c^* = c(s_i)$, $s^* = s_i$;
  // the FANT phase
  (3) initialize the memory structure
  (4) for $i = 1$ to $M$ do
  (4.1) construct a provisory solution $s$;
      (4.2) improve the solution $s$ by a local searcher
      (4.3) if $c(s) < c^*$ then
          (4.3.1) $s^* = s, c^* = c(s^*)$
          (4.3.2) initialize the memory structure
      (4.4) update the memory structure
  (5) return $s^*$;
End
```

To keep it simple and competitive, FANT differs from ACO in three fields: (1) Instead of using a swarm of artificial ants in standard ACO algorithms, FANT just use one ant to get solutions. (2) A local search procedure is added in FANT to improve the provisory solution obtained by the ant. Many works have shown that hybridization of local search and meta-heuristic framework is very effective, because local search is powerful in finding good solutions; while meta-heuristic like FANT can be efficient in jumping out of local optima. (3) No heuristic information is constructed in advance for FANT. Heuristic information makes the algorithm converge faster, and meanwhile makes the search get trapped in local optima with greater probability. Instead of using heuristic information, a diversification mechanism that reset the pheromone has been developed.

But the traditional FANT algorithms can't work well for the AP3, due to the fact that the solution space size of the AP3 is $(n!)^2$, which is far larger than other NP-hard problems, such as the quadratic assignment problem (QAP). However, we have observed in Section II that those triplets belonging to global optimal solutions are likely to frequently appear in local optimal solutions. Therefore, we can guide the ant by those triplets frequently appearing in local optimal solutions.

Upon the idea above, the sampling based FANT (S-FANT) is proposed for the AP3. As shown in Algorithm 2, the S-FANT consists of two phases. In the sampling phase, a multi-restart scheme is employed to generate local optimal solutions. After that, the pheromone is initialized according to the frequency of triplets appearing in those local optimal solutions. In the FANT phase, a standard FANT algorithm is conducted to explore for better solutions. Four components in the FANT are described in detail as follows.

A. Memory Structure

We use a three-dimensional matrix $prob_{n\times n\times n}$ as the pheromone structure, where each entry of the matrix $prob_{ijk}$ indicates the probability of choosing triplet (i, j, k).

```
Algorithm 3: Pheromone Initialization
Input: local optimal solutions $s_1, s_2, \cdots, s_k$, candidate size $m$
Output: $prob_{n\times n\times n}$
Begin
  (1) for each entry $prob_{ijk}$ of $prob_{n\times n\times n}$ do $prob_{ijk} = 0$;
  (2) for $t = 1$ to $k$ do
          for each triplet $(i,j,k) \in s_t$ do $prob_{ijk} = prob_{ijk} + 1$;
  (3) for every $i \in I$ do
      (3.1) sort all triplets $(i,j,k)$ $(j \in J, k \in K)$ by $prob_{ijk}$ in
      descending order
      (3.2) let $S$ be the set of $m$ triplets with highest
      entry values
      (3.3) for each triplet $(i,j,k) \notin S$ do $prob_{ijk} = 1$;
      (3.4) sort all entries $(i,j,k) \in S$ by $c_{ijk}$ in ascending
      order
      (3.5) reset the $prob_{ijk}$ for each triplet $(i,j,k) \in S$
End
```

Unlike traditional FANT algorithms, the initialization of our pheromone matrix $prob_{n\times n\times n}$ is given as Algorithm 3. Firstly, every entry of the pheromone matrix $prob_{n\times n\times n}$ is set to be 0 (see Step (1)). Secondly, the frequencies of triplets appearing in local optimal solutions are recorded as their pheromones (see Step (2)). Finally, the entries of the pheromone matrix are reset as follow. For every $i \in I$, triplets (i,j,k) $(j \in J, k \in K)$ are sorted by $prob_{ijk}$ in descending order (see Step (3.1)). After that, only m triplets with highest entry values will be chosen (Step (3.2)), while entry values of the remaining $n^2 - m$ triplets will be reset to be 1 (Step (3.3)), so as to guarantee efficiency of the construction of a provisory solution. Those chosen m triplets will then be resorted according to its cost so that the provisory solution will use triplets with smaller costs by greater probability (Step (3.4)). In our paper, the new value of $prob_{ijk}$ will be set to 300*(m+1−h) for every chosen triplet, where h represents the new triplet rank after the sorting (Step (3.5)).

B. Construction of a Provisory Solution

As shown before, provisory solutions are constructed based on the pheromone matrix (Algorithm 4). To generate a provisory solution, we use a constructive method that chooses the triplets successively, in a random order and with a probability proportional to the entry values contained in the matrix $prob_{n\times n\times n}$.

```
Algorithm 4: Provisory Solution Construction
Input: AP3 instance
Output: solution $s$
Begin
  (1) $I = J = K = \{1, 2, \cdots, n\}$, $s = \varnothing$;
  (2) for $t = 1$ to $n$ do
  (2.1) choose $i \in I$ randomly;
  (2.2) $I = I \setminus \{i\}$;
    (2.3) choose $j \in J$ randomly, uniformly with
    probability $\sum_{k\in K} prob_{ijk} / \sum_{j\in J} \sum_{k\in K} prob_{ijk}$;
  (2.4) $J = J \setminus \{j\}$;
    (2.5) choose $k \in K$ randomly, uniformly with
    probability $prob_{ijk} / \sum_{k\in K} prob_{ijk}$;
  (2.6) $K = K \setminus \{k\}$;
  (2.7) $s = s \bigcup \{(i, j, k)\}$;
End
```

C. Local Search

In each round of iteration, once the provisory solution is constructed, a local search procedure is employed to improve the solution. The local search procedure systematically search the neighborhood of the current solution, and this procedure works in an iterative fashion until no better solution can be found in the neighborhood of current solution. There're two local search procedures available in the literature: Hungarian local search and 2-Exchange local search.

- Hungarian local search is proposed by Huang and Lim (2006). The key idea of Hungarian local search is to map an AP3 to a series of linear assignment problems. Given a solution to AP3 consisting of two permutation p and q, if we fix permutation p, the optimization of q becomes an linear assignment problem which can be solved by Hungarian algorithm in $O(n^3)$. In a similar way, the permutation p and the index permutation of I can be optimized separately and efficiently (see Fig. 3).
- 2-Exchange local search optimizes the permutation p and q separately. Given a solution $s = \{(i, p(i), q(i))\}$ to AP3, it swap $p(i_1), p(i_2)$ for every pair of triplets $(i_1, p(i_1), q(i_1)), (i_2, p(i_2), q(i_2))$, if a better solution is obtained, the current solution is replaced with the new one. Such procedure repeats until no improvement occurs. Similar swap operation is also applied to the permutation q. An example of 2-Exchange is given in Fig. 4.

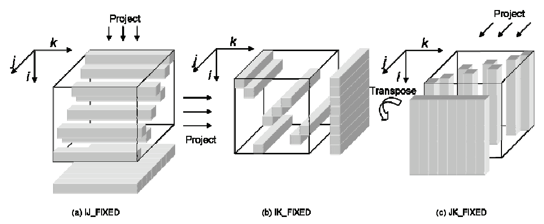
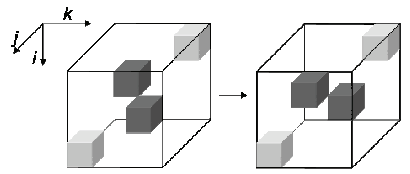
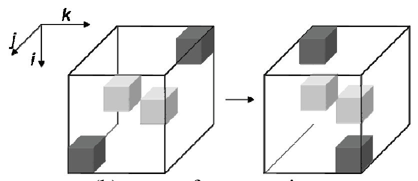


Fig. 3. Hungarian local search



(a) swap of permutation p



(b) swap of permutation q

Fig. 4. 2-Exchange local search.

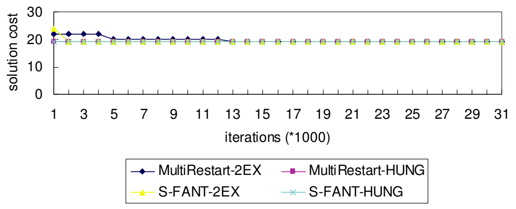
D. Memory Update

There are two parameters in the procedure of updating memory (pheromone): r and R, corresponding to the current solution constructed and the best solution, respectively. The entries of matrix $prob_{n\times n\times n}$ is updated as Algorithm 5.

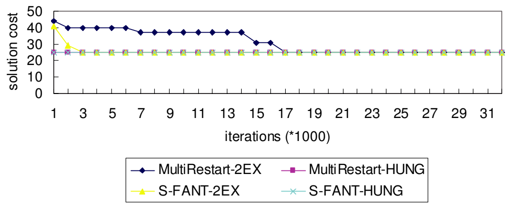
```
Algorithm 5: Pheromone Update
Input: current solution $s$, best solution $s^*$
Output: pheromone $prob_{n\times n\times n}$
Begin
  (1) for every triplet $(i, j, k) \in s$ do
      $prob_{ijk} = prob_{ijk} + r$;
  (2) for every triplet $(i, j, k) \in s^*$ do
      $prob_{ijk} = prob_{ijk} + R$;
End
```

IV. Experimental Results

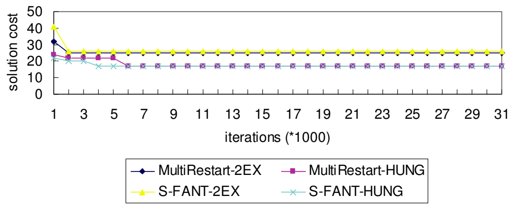
A. Convergence Analysis



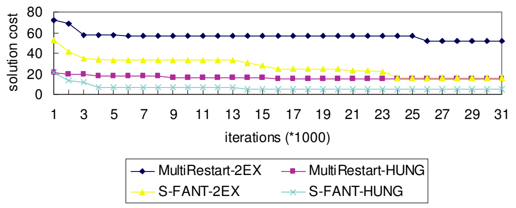
(a) bs_8_1



(b) bs_10_1



(c) bs_12_1



(d) bs_26_1

Fig. 5. Convergence speed.

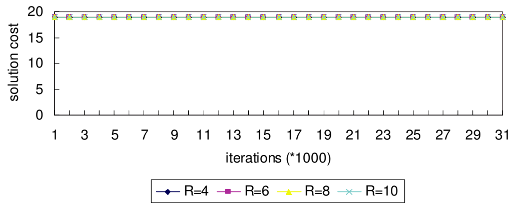
Fig. 5 shows the convergence results of S-FANT on the instance bs_8_1.dat, bs_10_1.dat, bs_12_1.dat and bs_26_1.dat from the Balas and Saltzman Dataset. As a comparison, a multi-restart local search algorithm is run on the same instances. In the experiments, we use 1000 local optimal solutions in the sampling phase of S-FANT and the iteration in the FANT phase is set to be 30000. As a comparison, the total iterations in MultiRestart is 31000. The parameter R is set to 4 in all the algorithms. Since there're two local search algorithms, we have two versions of S-FANT: S-FANT-2EX and S-FANT-HUNG (using 2-Exchange, Hungarian as the subordinate local search algorithm, respectively). Similarly, there're two versions of MultiRestart: MultiRestart-2EX and MultiRestart-HUNG.

The following remarks can be made from Fig. 5:

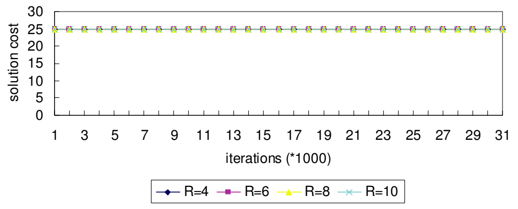
- MultiRestart-2EX is obviously bad in performance.
- S-FANT-HUNG is the best algorithms among those four algorithms. For all instances, S-FANT-HUNG can always find best solutions and converge faster than any other algorithm.
- Given any local search procedure (i.e., 2-Exchang, or Hungarian), S-FANT can always achieve better performance than MultiRestart.

Hence, it is evident that our S-FANT framework is successful. Guided by the pheromone, it is possible for local search to improve the solution quality consistently; and with the help of local search, the ant becomes competitive in finding good solutions.

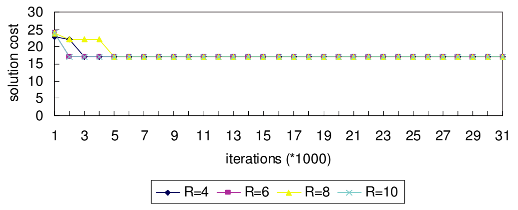
B. Parameter Tuning



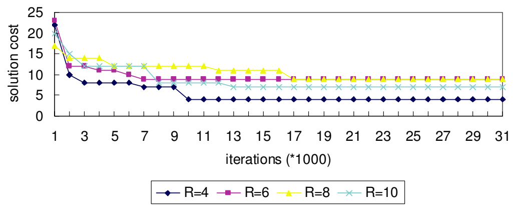
(a) bs_8_1



(b) bs_10_1



(c) bs_12_1



(d) bs_26_1

Fig. 6. Parameter tuning for R in S-FANT-HUNG.

We conducted some preliminary experiments to tune our algorithm because the parameter R can influence the performance of a FANT substantially. As shown in the above subsection, we use Hungarian local search as the subordinate local search procedure in S-FANT. For every instance, the iteration in the FANT phase is set to be 30000.

Fig. 6 shows the comparison between $R = 4$, $R = 6$, $R = 8$ and $R = 10$ on the instance bs_8_1.dat, bs_10_1.dat, bs_12_1.dat and bs_26_1.dat.

We can draw conclusions from Fig. 6:

- For small instances (bs_8_1.dat, bs_10_1.dat) the value of R won't affect the solution quality. The reason is that global optimal solutions can always be found in the sampling phase, due to the effectiveness of the Hungarian local search procedure.
- For larger instances (bs_26_1.dat), S-FANT-HUNG may achieve better performance when a small value is used for R.

Hence, in the following experiments, we set $R = 4$ for all instances.

C. *Computational Result*

In this subsection, we shall demonstrate the effectiveness of S-FANT by experimental results over the AP3 benchmark dataset. All the codes are implemented by C++ under a Pentium IV D2.8GHz with 1GB memory. For S-FANT, we use 1000 local optimal solutions in the sampling phase and 30000 iterations in the FANT phase.

Since the computing machine in Aiex's paper (2005) is a SGI Challenge R10000 machine, a scaling scheme is used according to SPEC (Standard Performance Evaluation Corporation, www.specbench.org/osg/cpu2000/) in order to compare the CPU time. It indicates that Pentium IV D2.8GHz is not more 14.3 times faster than SGI Challenge R10000 (see Appendix).

The AP3 dataset is generated by Balas and Saltzman (1991). It includes 60 test instances with the problem size n = 4, 6, 8, $\cdots$, 26. For each problem size n, five instances are randomly generated with the integer cost coefficients c_{ijk} uniformly distributed in the interval [0, 100].

Tab. 1 shows the results of our experiments on this dataset. Each row reports the average score of the five instances with the same size.

The column "Optimal'' shows the optimal solution reported by Balas and Saltzman, while column "B–S" is the result of their Variable Depth Interchange heuristic. Column "GRASP with Path Relinking" is the result reported in Aiex's paper. Column "MultiRestart-HUNG" is the result of MultiRestart using Hungarian local search. Finally, Column "S-FANT-HUNG" shows our algorithm using the Hungarian as the local search procedure. The best results among these algorithms are underlined in this table.

It is evident that our S-FANT can provide much better solutions than GRASP with Path Relinking. For instances whose size <=12, the MultiRestart-HUNG can always obtain global optimal solutions due to the effectiveness of Hungarian local search procedure. For instances whose size > 12, the sampling phase of S-FANT framework provides faster convergence speed so that S-FANT can obtain competitive solutions than the MultiRestart framework.

Tab. 1: Balas and Saltzman Dataset (12*5 instances)

n	Optimal	B-S	GRASP with Path Relinking			MultiRestart-HUNG		S-FANT-HUNG	
	Avg. Cost	Avg. Cost	Avg. Cost	Avg. CPU time (seconds)		Avg. Cost	Avg.CPU time (seconds)	Avg. Cost	Avg.CPU time (seconds)
				R10000	PIV D2.8G		PIV D2.8G		PIV D2.8G
4	42.2	43.2	-	-	-	42.2	0.18	42.2	0.21
6	40.2	45.4	-	-	-	40.2	0.61	40.2	1.26
8	23.8	33.6	-	-	-	23.8	1.22	23.8	1.53
10	19.0	40.8	-	-	-	19.0	2.04	19.0	1.88
12	15.6	24.0	15.6	74.79	>5.23	15.6	3.51	15.6	2.20
14	10.0	22.4	10.0	106.55	>7.45	12.6	3.95	10.0	4.40
16	10.0	25.0	10.2	143.89	>10.06	14.2	7.02	10.0	6.86
18	6.4	17.6	7.4	190.88	>13.35	12.6	9.99	8.2	9.78
20	4.8	27.4	6.4	246.70	>17.25	14.2	13.04	6.2	15.32
22	4.0	18.8	7.8	309.64	>21.65	17	16.17	7.6	17.31
24	1.8	14.0	7.4	382.45	>26.74	15	20.20	6.6	21.37
26	1.3	15.7	8.4	465.20	>32.53	15.2	24.88	7	23.96

V. Conclusions

In this paper, we proposed a sampling based FANT (S-FANT) for solving the AP3. The S-FANT consists of two phases: The sampling phase generates multiple local optimal solutions to help the ant converge faster in searching for global optimal solutions; In the FANT phase, a standard FANT algorithm is conducted to search for better solutions.

Work of this paper shows a promising way to design FANT algorithms for those NP-hard problems with large solution space, such as multi-objective problems. In the future work, we shall investigate the effectiveness of sampling phase in other ACO algorithms and design similar FANT algorithms for multi-objective problems.

APPENDIX

According to Tab. 2, Pentium IV D2.8G: R10000 = (1424 / 233) * (20.7 /8.85) = 14.2949 < 14.3.

Tab. 2: CPU Benchmark from SPEC

	Intel PIV D2.8G	Intel PIII 500	SGI Challenge R1000
SPECint 95		20.7	8.85
SPECint 2000	1424	233	

REFERENCES

[1] R.M. Aiex, M.G.C. Resende, P.M. Pardalos, G. Toraldo, "GRASP with Path Relinking for Three-Index Assignment," INFORMS Journal on Computing, 17(2), pp. 224-247, 2005.

[2] E. Balas, M.J. Saltzman, "An algorithm for the three-index assignment problem," Operations Research vol.39, pp. 150-161, 1991.

[3] K.D. Boese, "Cost versus distance in the traveling salesman problem," Technical Report CSD-950018, 1995.

[4] R.E. Burkard, R. Rudolf, "Three dimensional axial assignment problems with decomposable cost coefficients," Discrete Applied Mathematics vol.32, pp. 85-98, 1993.

[5] Y. Crama, A.W.J. Kolen, A.G.Oerlemans, F.C.R.Spieksma, "Throughput rate optimization in the automated assembly of printed circuit boards," Annals Operations Research, vol.26, pp. 455-480, 1990.

[6] Y. Crama, F.C.R. Spieksma, "Approximation algorithms for three-dimensional assignment problems with triangle inequalities," European Journal of Operations Research vol.60, pp. 273-279, 1992.

[7] A.M. Frieze, J. Yadegar, "An algorithm for solving 3-dimensional assignment problems with application to scheduling a teaching practice," Journal of Operations Research Society, vol.32, pp. 989-995, 1981.

[8] P. Hansen, L. Kaufman, "A primal-dual algorithm for the three-dimensional assignment problem," Cahiers du CERO vol.15, pp. 327-336, 1973.

[9] G. Huang, A. Lim, "A hybrid genetic algorithm for the Three-Index Assignment Problem," European Journal of Operational Research, 172(1), pp.249-257, 2006.

[10] P. Merz, B. Freisleben, "Fitness landscapes and memetic algorithms and greedy operators for graph bi-partitioning," Evolutionary Computation, 8(1), pp. 61-91, 2000.

[11] P.M. Pardalos, L.S. Pitsoulis, "Nonlinear assignment problems: algorithms and applications," Kluwer Academic Publishers, Boston, MA, 2000

[12] W.P. Pierskalla, "The tri-substitution method for the three-dimensional assignment problem," Canadian Operations Research Society Journal, vol.5, pp. 71-81, 1967.

[13] W.P. Pierskalla, "The multidimensional assignment problem," Operations Research, vol.16, pp. 422-431, 1968.

[14] C.R. Reeves, "Landscapes, operators and heuristic search," Annals of Operation Research, 86(1), pp. 473-490, 1999.

[15] J. Schneider, "Searching for backbones-a high-performance parallel algorithm for solving combinatorial optimization problems," Future Generation Computer Systems, 19(1), pp.121-131, 2003.

[16] E.D. Taillard, L. Gambardella, "Adaptive memories for the quadratic assignment problems," Technical Report, IDSIA-87-97, Lugano: IDSIA, 1997.

[17] S. Voss, "Heuristics for nonlinear assignment problems," P.M. Pardalos, L.S. Pitsoulis, eds. Nonlinear Assignment Problems: algorithms and applications. Kluwer Academic Publishers, Boston, MA, pp. 175-215, 2000.

[18] P. Zou, Z. Zhou, G.L. Chen, H. Jiang, J. Gu, "Approximate-backbone guided fast ant algorithms to QAP," Journal of Software 16(10), pp. 1691-1698, 2005.