Adaptive Tabu Search for Traveling Salesman Problems

S. Suwannarongsri and D. Puangdownreong

***Abstract*—**One of the most intensively studied problems in computational mathematics and combinatorial optimization is the traveling salesman problem (TSP). The TSP is classified and considered as the class of the NP-complete combinatorial optimization problems. By literatures, many algorithms and approaches have been launched to solve such the TSP. However, no current algorithms can provide the exactly optimal solution of the TSP problem. This article proposes the application of adaptive tabu search (ATS), one of the most powerful AI search techniques, to solve the TSP problems. The ATS is tested against ten benchmark real-world TSP problems. Results obtained by the ATS will be compared with those obtained by the genetic algorithms (GA) and the tabu search (TS). As results, the ATS, TS, and GA can provide very satisfactory solutions for all TSP problems. Among them, the ATS outperforms other algorithms.

***Keywords*—**Adaptive tabu search, genetic algorithm, tabu search, traveling salesman problem.

I. Introduction

The traveling salesman problem (TSP) has been firstly proposed as one of the mathematical problems for optimization in 1930s [1]. The problem is to find an optimal tour for a traveling salesman wishing to visit each of a list of n cities exactly once and then return to the home city. Such optimal tour is defined to be a tour whose total distance (cost) is minimized. This problem can be considered as the class of combinatorial optimization problems known as NP-complete [1], [2]. By literature, many algorithms and approaches have been launched to solve the TSP problems. Those algorithms and approaches can be classified into exact and heuristic approaches [3], [4], [5].

The TSP problems possessing no longer than 20 cities can be optimally solved by exact methods. Some are dynamic programming [6], branch and bound [7], and linear programming [2]. The heuristic methods could provide very satisfactory solutions of the TSP problems possessing large amount number of cities. However, the optimum solution can not be guaranteed. To date, the artificial intelligent (AI) search techniques have been applied to solve the TPS problems, for example, simulated annealing (SA) [8], artificial neural network (ANN) [9], tabu search (TS) [10], and genetic algorithms (GA) [11].

By literatures, the adaptive tabu search (ATS), one of the most efficient and powerful AI search techniques, has been launched since 2002 [12], [13]. The ATS is widely applied to various engineering applications such as system and model identification [14], system performance optimization [15], assembly line balancing problem [16], surface optimization [17], and control synthesis [18]. In this paper, the ATS is then applied to solve ten benchmark real-world TSP problems collected in TSPLIB95 [19]. Based on exactly optimal solutions, results obtained by the ATS will be compared with those obtained by two well-known AI search techniques, i.e. the GA and TS.

This article consists of five sections. After an introduction provided in section 1, the problem formulation of TSP problem optimization is formulated in section 2. The ATS algorithms as well as TS and GA are briefly described in section 3. AI-based TSP solving is given in section 4, while conclusions are provided in Section 5.

II. TSP Problem Formulation

By theory, the traveling salesman problem (TSP) has been firstly proposed as one of the mathematical problems in 1800s by Harmilton and Kirkman. However, the general formulation of TSP has been firstly lunched based on the graph theory in 1930s [1].

Let $G = (V, E)$ be a complete undirected graph with vertices V, $|V| = n$, where n is the number of cities, and edges E with edge length c_{ij} for the-ij city (i, j). Our work focus on the symmetric TSP case in which $c_{ij} = c_{ji}$, for all cities (i, j). The TSP problem formulations for minimization as expressed in (1) – (5) [2]. Equation (1) is the objective function, which minimizes the total distance to be traveled. Constraints (2) and (3) define a regular assignment problem, where (2) ensures that each city is entered from only one other city, while (3) ensures that each city is only departed to on other city. Constraint (4) eliminates subtours. Constraint (5) is a binary constraint, where $x_{ij} = 1$ if edge (i, j) in the solution and $x_{ij} = 0$, otherwise.

Manuscript received February 26, 2012: Revised version received February 26, 2012.

S. Suwannarongsri is with Department of Industrial Engineering, Faculty of Engineering, South-East Asia University, Bangkok, Thailand, 10160 (corresponding author phone: 66-2807-4500; fax: 66-2807-4528; e-mail: supaporns@sau.ac.th).

D. Puangdownreong is with Department of Electrical Engineering, Faculty of Engineering, South-East Asia University, Bangkok, Thailand, 10160 (e-mail: deachap@sau.ac.th).

$$\text{Min} \quad \sum_{i\in V}\sum_{j\in V} c_{ij}x_{ij} \tag{1}$$

$$\text{Subject to} \quad \sum_{\substack{j\in V \\ j\neq i}} x_{ij} = 1, \quad i \in V \tag{2}$$

$$\sum_{\substack{i\in V \\ i\neq j}} x_{ij} = 1, \quad j \in V \tag{3}$$

$$\sum_{i\in S}\sum_{j\in S} x_{ij} \leq |S| - 1, \quad \forall S \subset V, S = \varnothing \tag{4}$$

$$x_{ij} = 0 \text{ or } 1, \quad i, j \in V \tag{5}$$

However, the difficulty of solving TSP is that subtour constraints will grow exponentially as the number of city grows large, so it is not possible to generate or store these constraints. Many applications in real-world do not demand optimal solutions. Therefore, many researchers proposed several heuristic algorithms, which are fast and easy to implement.

III. AI Algorithms

The artificial intelligent (AI) search techniques used to solve the TSP problems in this work are the genetic algorithms (GA), the tabu search (TS), and the adaptive tabu search (ATS). Their algorithms are briefly reviewed as follows.

A. Genetic Algorithm

The genetic algorithm or GA is one of AI search optimization techniques. GA has natural selection mechanism and genetic operation, i.e. crossover and mutation techniques to find optimum solution. The GA algorithms can be briefly summarized as follows [20], [21].

Step 1. Randomly generate the populations.
Step 2. Evaluate all population chromosomes via the objective function.
Step 3. Select some chromosomes and set them as parents.
Step 4. Generate next generation of population by crossover and mutation.
Step 5. Evaluate the (fitness) objective function of new populations.
Step 6. Replace old population by new ones that more fit.
Step 7. Once termination criteria (TC) are met, terminate search process; otherwise go back to Step 2.

GA will stop the search process once the TC is satisfied. Generally, we use the preset maximum generation ($gen_{\max}$) and the maximum allowance of the global solution ($J_{\max}$) as the TC. This implies that the GA search process will be stopped, once $gen = gen_{\max}$ or the current solution is less than $J_{\max}$. The optimum solution is the best chromosome in current population.

B. Tabu Search

The tabu search or TS is proposed by Glover [22], [23]. The TS is also one of AI search optimization techniques. Based on the neighborhood search, the TS has the tabu list (TS) used to store the visited solutions and to conduct as an aspiration criteria when the local entrapment occurs. The TS algorithms can be briefly described as follows [24], [25].

Step 1. Initialize a search space (Ω), TL = $\varnothing$, search radius (R), *count*, and $count_{\max}$.
Step 2. Randomly select an initial solution S_0 from a certain search space Ω. Let S_0 be a current local minimum.
Step 3. Randomly generate N solutions around S_0 within a search radius R. Store the N solutions, called neighborhood, in a set X.
Step 4. Evaluate the objective value of each member in X via objective functions. Set S_1 as a member giving the minimum cost.
Step 5. If $f(S_1) < f(S_0)$, put S_0 into the TL and set $S_0=S_1$, otherwise, store S_1 in the TL instead.
Step 6. If the TC : $count = count_{\max}$ or desired specification are met, then stop the search process. S_0 is the best solution, otherwise Update $count = count+1$, and go back to Step 2.

Like GA, TS will stop the search process once the TC is satisfied. Generally, we use the preset maximum search round or count ($count_{\max}$) and the maximum allowance of the global solution ($J_{\max}$) as the TC. This means that the TS search process will be stopped, once $count = count_{\max}$ or the current solution is less than $J_{\max}$.

C. Adaptive Tabu Search

The adaptive tabu search or ATS is the modified version of the TS. Based on iterative neighborhood search approach, the ATS was launched in 2004 [12], [13]. The ATS search process begins the search with some random initial solutions belonging to a neighborhood search space. All solutions in neighborhood search space will be evaluated via the objective function. The solution giving the minimum objective cost is set as a new starting point of next search round and kept in the tabu list (TL). Fig. 1 illustrates some movements of the ATS. The ATS possesses two distinctive mechanisms denoted as back-tracking (BT) regarded as one of the diversification strategies and adaptive radius (AR) considered as one of the intensification strategies. The ATS can be regarded as one of the most powerful AI search techniques. Convergence proof and performance evaluation of the ATS have been reported [12], [13]. The ATS algorithm is summarized step-by-step as follows.

Step 1. Initialize a search space (Ω), TL = $\varnothing$, search radius (R), *count*, and $count_{\max}$.
Step 2. Randomly select an initial solution S_0 from a certain search space Ω. Let S_0 be a current local minimum.

Step 3. Randomly generate N solutions around S_0 within a search radius R. Store the N solutions, called neighborhood, in a set X.

Step 4. Evaluate the objective value of each member in X via objective functions. Set S_1 as a member giving the minimum cost.

Step 5. If $f(S_1) < f(S_0)$, put S_0 into the TL and set $S_0=S_1$, otherwise, store S_1 in the TL instead.

Step 6. Activate the BT mechanism, when a local entrapment occurs.

Step 7. If the TC : $count = count_{max}$ or desired specification are met, then stop the search process. S_0 is the best solution, otherwise go to Step 8.

Step 8. Invoke the AR mechanism, once the search approaches the local or the global solution to refine searching accuracy.

Step 9. Update $count = count+1$, and go back to Step 2.

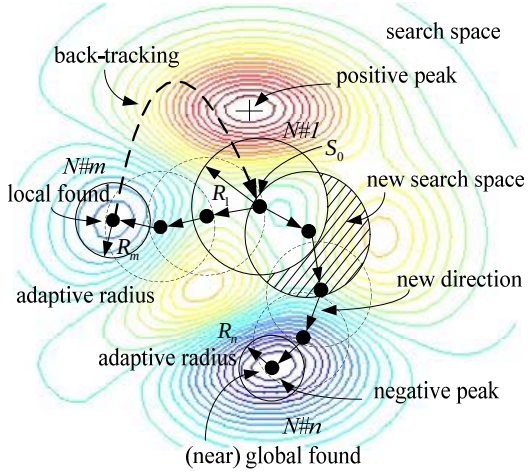


Fig. 1 some movements of ATS

The diagram in Fig. 2 reveals the search process of the ATS algorithm. The BT mechanism described in Step 6 is active when the number of solution cycling is equal to the maximum solution-cycling allowance. This mechanism selects an already visited solution stored in the TL as an initial solution for the next search round to enable a new search path that could escape the local deadlock towards a new local minimum. For the AR mechanism described in Step 8, it is invoked when a current solution is relatively close to a local minimum. The radius is thus decreased in accordance with the best cost function found so far. The less the cost function, the smaller the radius. With these two features, a sequence of solutions obtained by the ATS method rapidly converges to the global minimum. Like TS, ATS will stop the search process once the TC is satisfied. This means that the ATS search process will be stopped, once $count = count_{max}$ or the current solution is less than J_{max}. The following recommendations found in [12], [13] are useful for setting the initial values of search parameters:

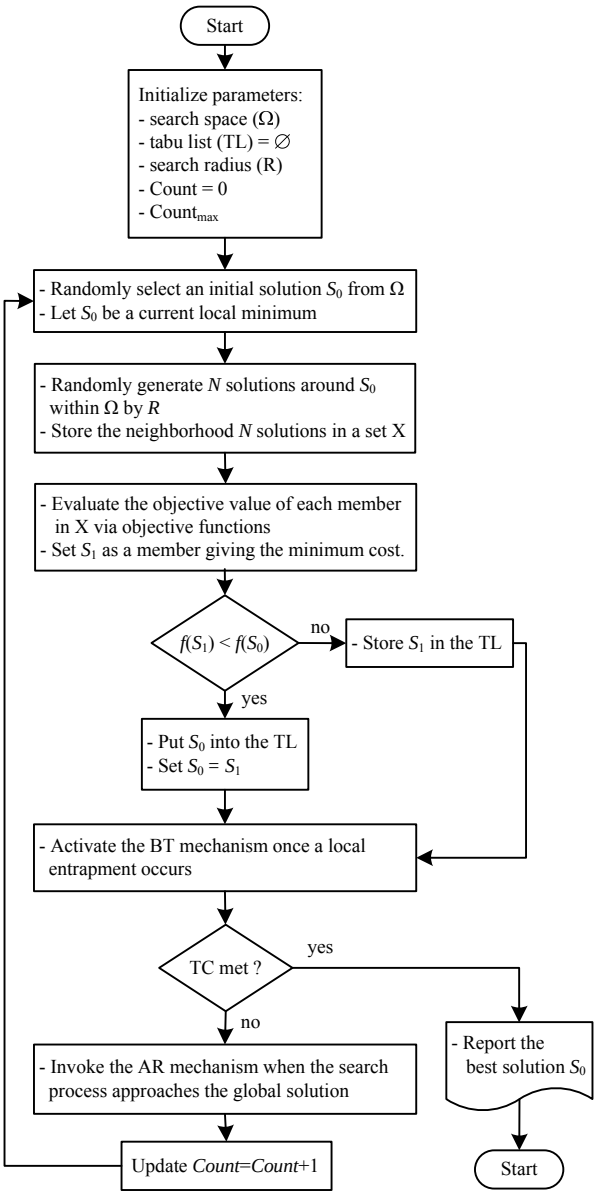


Fig. 2 flow diagram of ATS algorithm

(i) the initial search radius, R, should be 7.5-15.0% of the search space radius,

(ii) the number of neighborhood members, N, should be 30-40,

(iii) the number of repetitions of a solution before invoking the back-tracking mechanism should be 5-15,

(iv) the k_{th} backward solution selected by the back-tracking mechanism should be equal or close to the number of repetitions of a solution before invoking the back-tracking mechanism,

(v) the adaptive search radius should employ 20-25% of radius reduction, and

(vi) a well educated guess of the search space that is wide enough to cover the global solution is necessary.

IV. AI-BASED TSP SOLVING

In this work, algorithms of the GA, TS, and ATS are coded by MATLAB running on Pentium (R), 2.00 GHz CPU, 1 GB RAM, to solve ten benchmark real-world TSP problems collected in TSPLIB95 [19]. Details of selected benchmark TSP problems are summarized in Table 1. Each problem will be tested over 20 times to calculate the average optimum distance and the average search time. For a fair comparison, the search parameters of these AI algorithms will be a priory set as follows.

For GA :
- number of population = 100,
- crossover = 70%,
- mutation = 4.5%,
- replacement with 1 point, and
- TC : maximum generation = 2,000.

For TS :
- Ω = [1, 2, …, No. of cities],
- number of neighborhood members N = 100,
- search radius R = 20% of Ω, and
- TC : $count_{max}$ = 2,000.

Table 1 selected real-world TSP problems

TSP Problems	No. of Cities	Opt. distance (km.)
Eil51	51	426
Berlin52	52	7,542
St70	70	675
Pr76	76	108,159
Eil76	76	538
Rat99	99	1,211
Rd100	100	7,910
KroA100	100	21,282
KroB100	100	22,141
Ch150	150	6,528

For ATS :
- Ω = [1, 2, …, No. of cities],
- number of neighborhood members N = 100,
- search radius R = 20% of Ω,
- Activate BT, when local entrapment occurs,
- invoke AR once 20, 40, 60, and 80 times of solution cannot be improved, and
- TC : $count_{max}$ = 2,000.

Results obtained are summarized in Table 2. We found that GA, TS, and ATS can fine the optimum distant of all problems. From the average optimum distance in Table 2, the ATS outperforms other algorithms. The second is the GA, and the third is the TS, respectively. This may because the search process of the TS and the GA hit many local entrapments, while the ATS can efficiently escape such the local entrapments. Convergent curves of the cost function obtained by the GA, TS, and ATS over the Eil51 problem are depicted in Fig. 3 as an example. The convergence curves of other problems are omitted because they have a similar form to those of the Eil51. Figs. 4 – 13 depict the results of all TSP problems obtained by the GA, TS, and ATS, respectively.

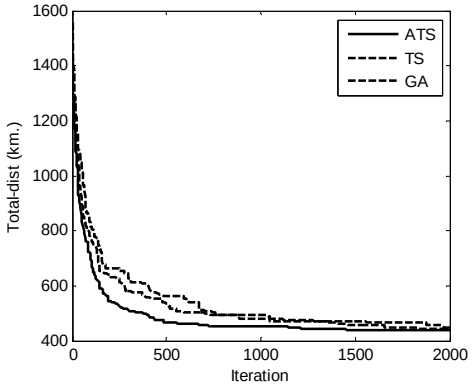
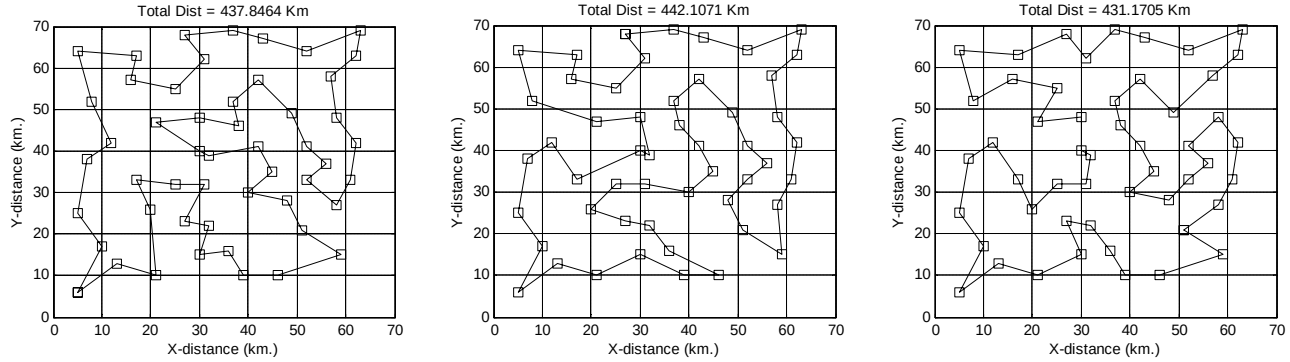


Fig. 3 convergent rate of Eil51

Table 2 results obtained by AI search techniques

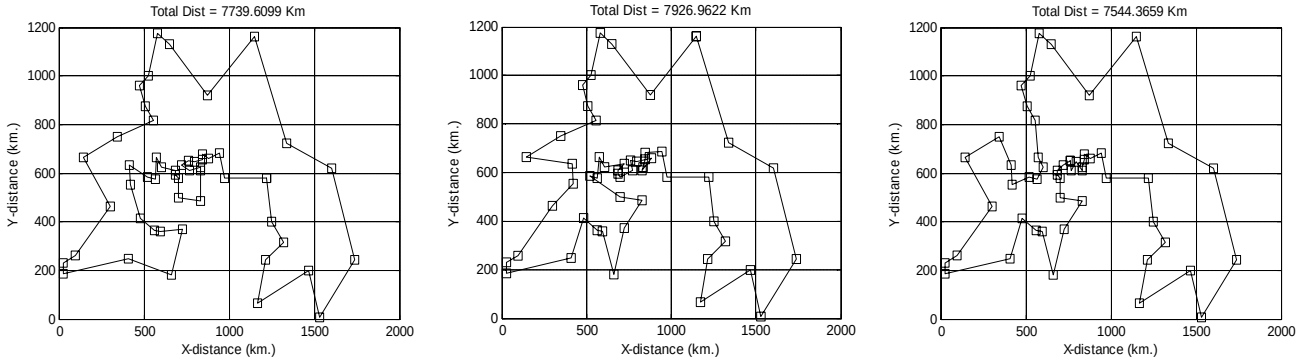
TSP problems	Opt. distance (km.)	Obtained solutions (average distance (Km.)) by AI techniques						average search time (sec.) by AI		
		GA	%Err	TS	%Err	ATS	%Err	GA	TS	ATS
Eil51	426	441.46	3.63	445.05	4.47	438.12	2.85	2.72	2.53	2.91
Berlin52	7,542	7,833.26	3.86	8,152.79	8.10	7,702.16	2.12	3.58	2.15	4.14
St70	675	695.44	3.03	738.78	9.45	684.62	1.43	3.84	3.19	4.22
Pr76	108,159	116,273.12	7.50	123,240.57	13.94	110,478.35	2.14	4.13	4.05	4.57
Eil76	538	548.26	1.91	582.62	8.29	540.17	0.40	4.05	3.86	4.43
Rat99	1,211	1,274.84	5.27	1,295.42	6.97	1,213.50	0.21	6.28	5.83	7.35
Rd100	7,910	8,506.51	7.54	8,788.39	11.10	8,412.62	6.35	10.68	9.12	13.46
KroA100	21,282	22,875.42	7.49	23,644.18	11.10	21,525.56	1.14	11.97	10.89	15.13
KroB100	22,141	24,765.94	11.86	25,349.36	14.49	22,568.14	1.93	12.52	11.18	15.46
Ch150	6,528	6,986.35	7.02	7,012.21	7.42	6,612.74	1.30	14.84	12.57	16.73

Note : %Err stands for percentage of solution errors compared with optimum distances.



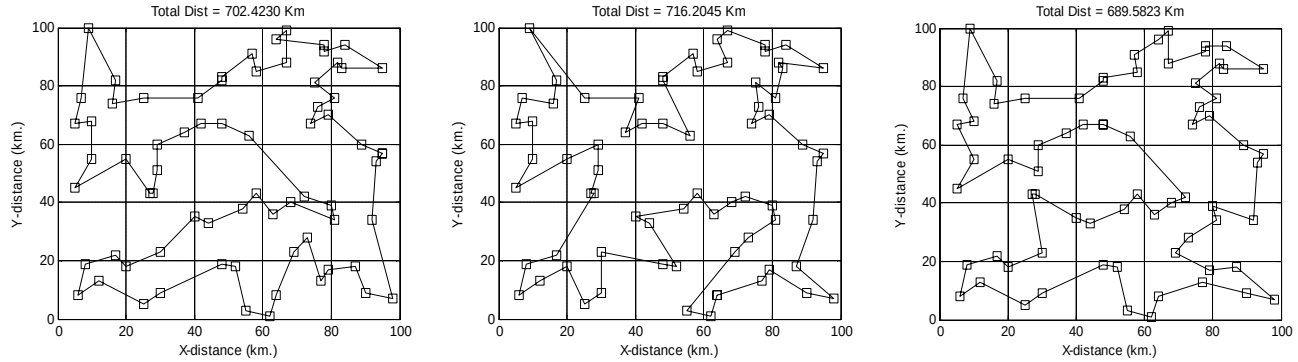
(a) (b) (c)

Fig. 4 results of Eil51 obtained by (a) GA (b) TS (c) ATS



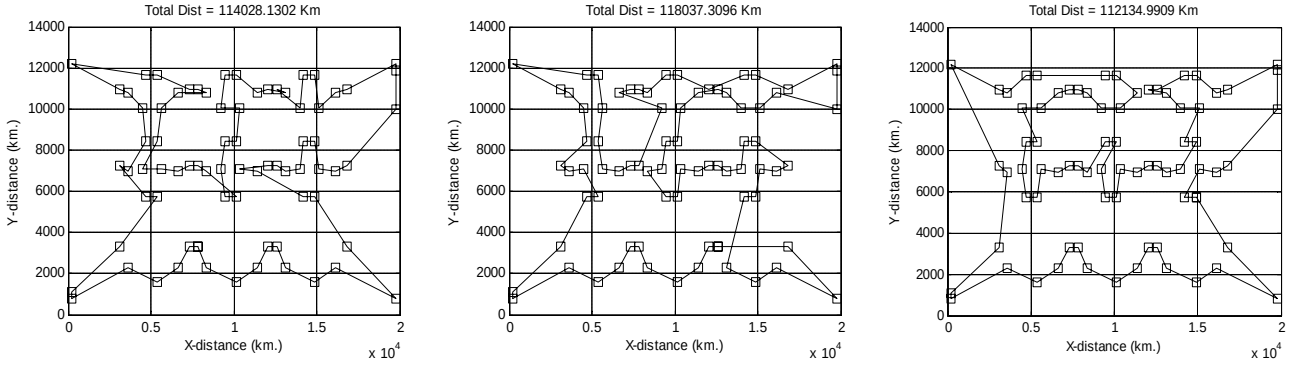
(a) (b) (c)

Fig. 5 results of Berlin52 obtained by (a) GA (b) TS (c) ATS



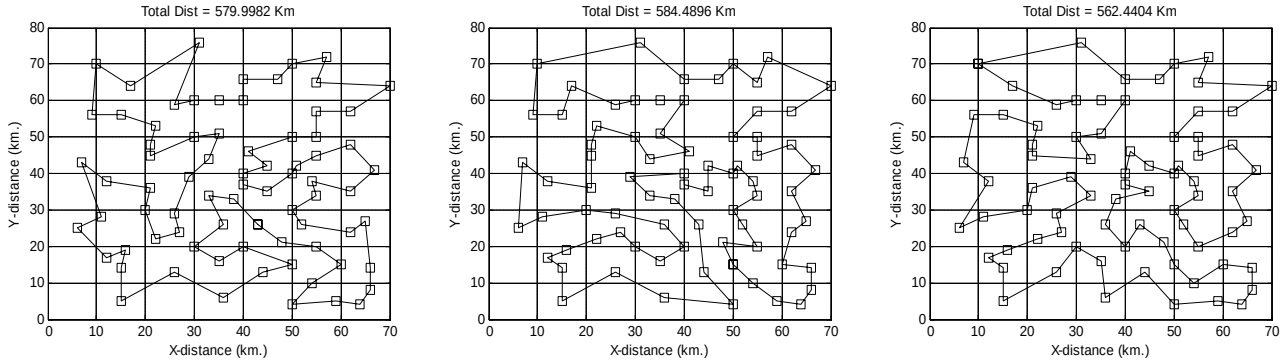
(a) (b) (c)

Fig. 6 results of St70 obtained by (a) GA (b) TS (c) ATS



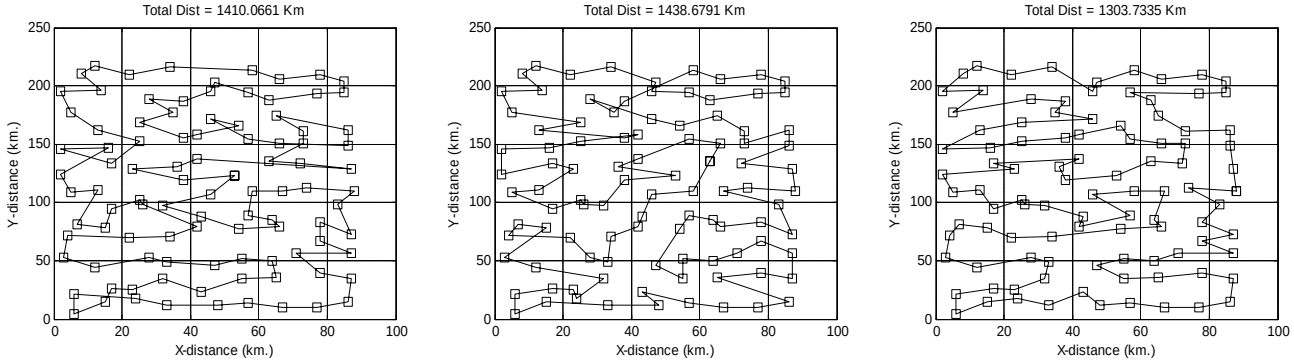
(a) (b) (c)

Fig. 7 results of Pr76 obtained by (a) GA (b) TS (c) ATS



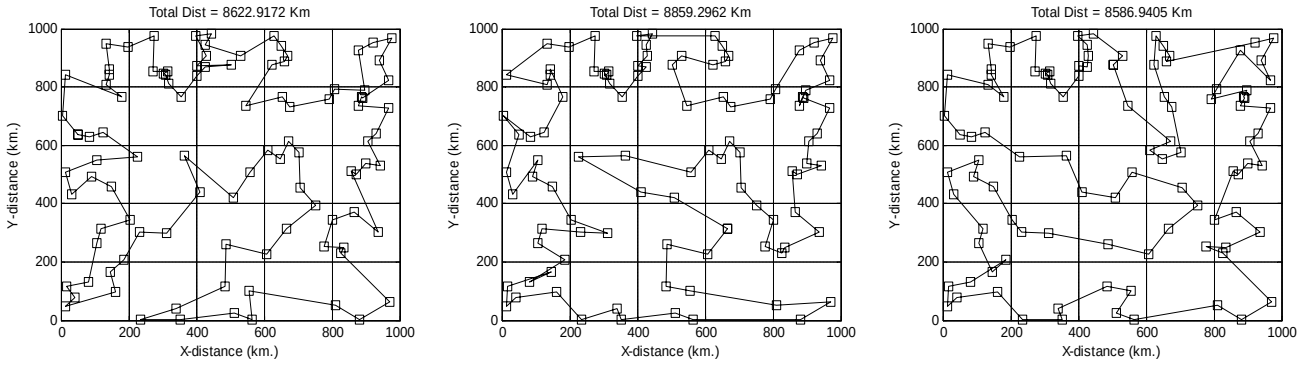
(a) (b) (c)

Fig. 8 results of Eil76 obtained by (a) GA (b) TS (c) ATS



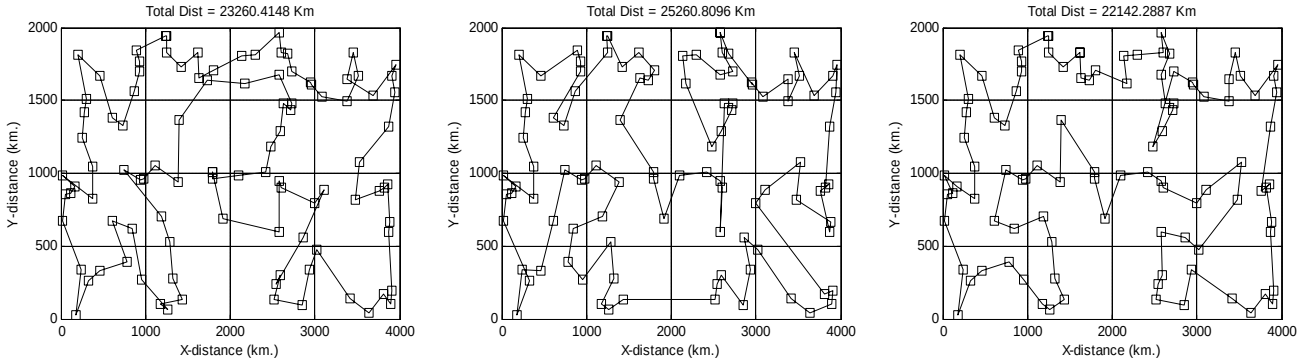
(a) (b) (c)

Fig. 9 results of Rat99 obtained by (a) GA (b) TS (c) ATS



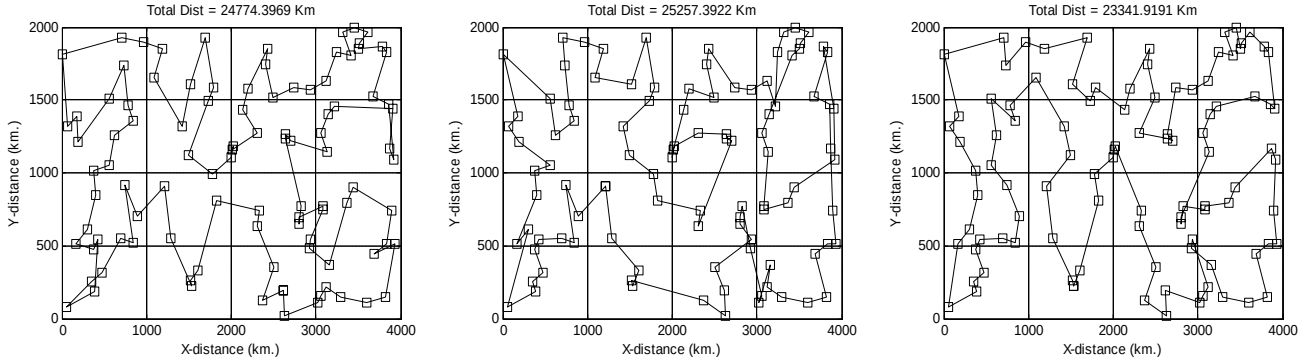
(a) (b) (c)

Fig. 10 results of Rd100 obtained by (a) GA (b) TS (c) ATS



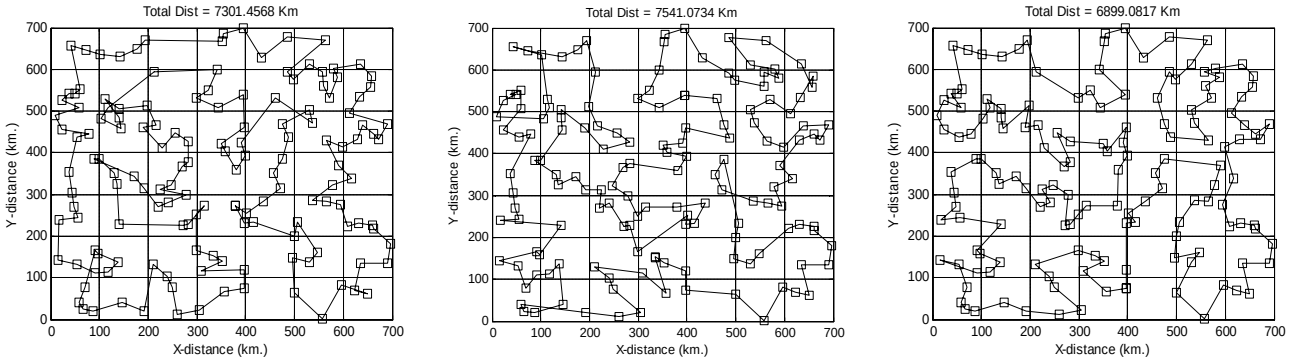
(a) (b) (c)

Fig. 11 results of KroA100 obtained by (a) GA (b) TS (c) ATS



(a) (b) (c)

Fig. 12 results of KroB100 obtained by (a) GA (b) TS (c) ATS



(a) (b) (c)

Fig. 13 results of Ch150 obtained by (a) GA (b) TS (c) ATS

In Table 2, %Err standing for percentage of solution errors compared with optimum distances are represented by the bar graph shown in Fig. 14. This can be noticed that the ATS can provide better solutions than the GA and TS, although it spends larger search time than other do as shown in Fig. 15.

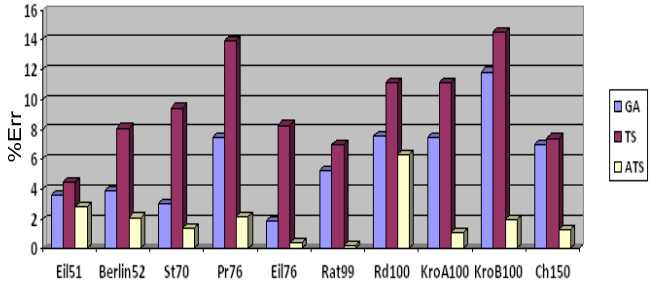


Fig. 14 %Err obtained by AI search techniques

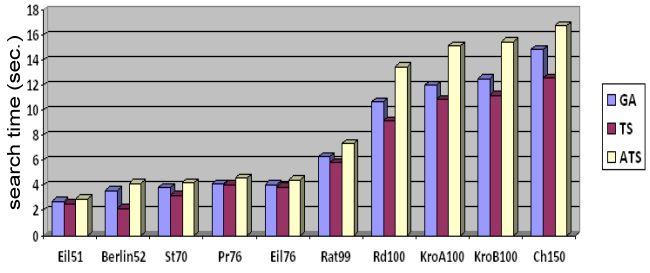


Fig. 15 average search times by AI search techniques

V. Conclusion

Solving the traveling salesman problem (TSP) by AI search techniques has been proposed in this article. The adaptive tabu search (ATS), one of the most powerful AI search techniques, has been applied to solve the TSP problems. It has been tested against ten benchmark real-world TSP problems collected in TSPLIB95. Based on the exactly optimal solutions, results obtained by the ATS would be compared with those obtained by the genetic algorithms (GA) and the tabu search (TS). As results, the ATS, TS, and GA can provide very satisfactory solutions for all TSP problems. Among them, the ATS can provide the best solution within reasonable search time consumed. This can be concluded that ATS outperforms other algorithms used in this article.

References

[1] M. Bellmore and G. L. Nemhauser, "The traveling salesman problem: a survey", *Operation Research*, vol. 16, 1986, pp.538-558.

[2] G. B. Dantzig, D. R. Fulkerson, and S. M. Johnson, "Solution of a large-scale traveling salesman problem", *Operation Research*, vol. 2, 1954, pp.393.

[3] G. Reinelt, *The Traveling Salesman : Computational Solutions for TSP Applications*. Springer-Verlag, 1994.

[4] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, *The Traveling Salesman Problem : A Guided Tour of Combinatorial Optimization*, John-Wiley & Sons, 1986.

[5] D. S. Johnson and L. A. McGeoch, "The traveling salesman problem: a case study in local optimization", *Local Search in Combinatorial Optimization*, Wiley, 1997, pp.215-310.

[6] M. Held and R. Karp, "A dynamic programming approach to sequencing problems", *SIAM J*, 1962, pp.196.
[7] J. Little, K. Murty, D. Sweeney, and C. Karel, "An algorithm for the traveling salesman problem", *Operation Research*, vol. 12, 1963, pp.972.
[8] E. H. L. Aarts, J. H. M. Korst, and P. J. M. Laarhoven, "A quantitative analysis of the simulated annealing algorithm: a case study for the traveling salesman problem", *J. of Stats Phys*, vol. 50, 1988, pp.189-206.
[9] J. V. Potvin, "The traveling salesman problem: a neural network perspective", *INFORMS J. of Computing*, vol. 5, 1993, pp. 328-348.
[10] C. N. Fiechter, "A parallel tabu search algorithm for large scale traveling salesman problems", *Discrete Applied Mathematics*, vol. 51(3), 1994, pp. 243-267.
[11] J. V. Potvin, "Genetic algorithms for the traveling salesman problem", *Annuals of Operation Research*, vol. 63, 1996, pp.339-370.
[12] D. Puangdownreong, T. Kulworawanichpong, and S. Sujitjorn, "Finite convergence and performance evaluation of adaptive tabu search", *LNCS* (*Lecture Notes in Computer Science*), Springer-Verlag Heidelberg, vol. 3215, 2004, pp.710-717.
[13] S. Sujitjorn, T. Kulworawanichpong, D. Puangdownreong, and K-N. Areerak, "Adaptive tabu search and applications in engineering design", *Book Chapter in Integrated Intelligent Systems for Engineering Design* (*ed. X.F. Zha and R.J. Howlett*), IOS Press, Netherlands, 2006, pp.233-257.
[14] D. Puangdownreong and S. Sujitjorn, "Image approach to system identification," *WSEAS Transactions on Systems*, vol. 5(5), 2006, pp.930-938.
[15] D. Puangdownreong, C. U-Thaiwasin, and S. Sujitjorn, "Optimized performance of a 2-mass rotary system using adaptive tabu search," *WSEAS Transactions on Circuits and Systems*, vol. 5(3), 2006, pp.339-345.
[16] S. Suwannarongsri and D. Puangdownreong, "Metaheuristic approach to assembly line balancing," *WSEAS Transactions on Systems*, vol. 8(2), 2009, pp.200-209.
[17] J. Kluabwang, D. Puangdownreong and S. Sujitjorn, "Management agent for search algorithms with surface optimization applications," *WSEAS Transactions on Computers*, vol. 7(6), 2008, pp.791-803.
[18] J. Kluabwang, D. Puangdownreong and S. Sujitjorn, "Performance assessment of search management agent under asymmetrical problems and control design applications," *WSEAS Transactions on Computers*, vol. 8(4), 2009, pp.691-704.
[19] TSPLIB95, *Symmetric traveling salesman problem*, http://comopt.ifi.uni-heidelberg.de/ software/TSPLIB95/. 5 February, 2010.
[20] H. Holland, *Adaptation in Natural and Artificial Systems*, Ann Arbor, The University of Michigan Press, 1975.
[21] D. E. Goldberg, *Genetic Algorithm in Search, Optimization, and Machine Learning*, Addison-Wesley Publishing, 1989.
[22] F. Glover, "Future paths for integer programming and links to artificial intelligence", *Computers and Operations Research*, vol. 13, 1986, pp.533-549.
[23] F. Glover, "Tabu search – part I", *ORSA Journal on Computing*, vol.1, 1989, pp.190-206.
[24] F. Glover, "Tabu search – part II", *ORSA Journal on Computing*, vol.2, 1990, pp.4-32.
[25] F. Glover, "Tabu search for nonlinear and parametric optimization (with links to genetic algorithms)", *Discrete Applied Mathematics*, vol. 49, 1994, pp.231-255.

Supaporn Suwannarongsri was born in Bangkok, Thailand, in 1983. She received the B.Eng. degree in industrial engineering from Thonburi University (TRU), Bangkok, Thailand, in 2004 and the M.Eng. degree in industrial engineering from King Mongkut's Institute of Technology Ladkrabang (KMITL), Bangkok, Thailand, in 2008, respectively.

Since 2006, she has been with the Department of Industrial Electrical Engineering, Faculty of Engineering, South-East Asia University (SAU), Bangkok, Thailand, where she is currently an assistant professor of industrial engineering and Head of Department of Industrial Engineering. Her research interests include operation research, production planning and design, and applications of AI search algorithms to various real-world industrial engineering problems.

Deacha Puangdownreong was born in Pranakhonsri Ayutthaya, Thailand, in 1970. He received the B.Eng. degree in electrical engineering from South-East Asia University (SAU), Bangkok, Thailand, in 1993, the M.Eng. degree in control engineering from King Mongkut's Institute of Technology Ladkrabang (KMITL), Bangkok, Thailand, in 1996, and the Ph.D. degree in electrical engineering from Suranaree University of Technology (SUT), Nakhon Ratchasima, Thailand, in 2005, respectively.

Since 1994, he has been with the Department of Electrical Engineering, Faculty of Engineering, South-East Asia University, where he is currently an associated professor of electrical engineering. While remains active in research, he served the university under various administrative positions including Head of Department of Electrical Engineering, Deputy Dean of Faculty of Engineering, Director of Research Office, and Director of Master of Engineering Program. His research interests include control synthesis and identification, AI and search algorithms as well as their applications. He has authored 3 books and published as authors and coauthors of more than 70 research and technical articles in peer-reviewed journals and conference proceedings nationally and internationally.

Dr.Deacha has been listed in Marquis Who's Who in the World, Marquis Who's Who in Science and Engineering, and Top 100 Engineers-2011 in International Biographical Center, Cambridge, UK.