Traveling-Salesman Competition with Multiple Agents and Simultaneous Move Search Algorithm

Nobuhiro Tomabechi
Department of System and Information Engineering
Hachinohe Institute of Technology
Hachinohe, Japan
tomabech@hi-tech.ac.jp

Yoshichika Fujioka
Department of System and Information Engineering
Hachinohe Institute of Technology
Hachinohe, Japan

Abstract—**In this paper, we study a traveling-salesman type competition in which several players travel around various objects distributed in a given area and compete with each other in collecting the objects. Similar competitions can be widely found in the real world such as in selling competition, in the competitive gathering of scraps, wastes, seafood or wild foods, etc. In this competition, players move simultaneously and the number of players can be more than two. We present a novel scheduling algorithm suitable for the competition in which a move toward an object along the shortest path is taken as the basic move, and the move is stopped halfway when opponents have the advantage. In the algorithm, the moves of the players can be dealt with quasi-alternately. However, a player can make several consecutive moves, and a move already made can be aborted. A position evaluation method based on the concept of territory is also presented. A simulation program is developed and competitions are held among computers or between a computer and a human being. It is confirmed that the algorithm and the position evaluation method can be effectively applied to the game model.**

Index Terms—**traveling-salesman, competition, multi-agent, search, algorithm, simultaneous**

I. Introduction

The traveling-salesman problem is one of the most intensively studied subjects in computer science [1],[2]. In this paper, we discuss a traveling-salesman type competition in which several salesmen independently visit n cities and compete with each other for the number of visiting cities [3],[4].

Similar competitions can be widely found in the real world such as in selling competitions, in transportation jobs involving competitors, and in the competitive gathering of scraps, wastes, seafood or wild foods, etc. Hence the competition discussed here will be a typical subject in computer science.

Concerning the scheduling problems of multiple agents, a variety of studies have been reported [5]-[8], however few reports on this subject have been found.

The competition can be modeled as a game such that several players travel around various objects distributed in a given area and compete with each other in collecting the objects. We assume that all the information about objects and players can be known at any time, that is, the game model is a complete-information game. The number of players can be more than two, that is, arbitrary. It is also possible to extend the game to team games in which the members of a team cooperate with each other and compete against the opposing team. However, the results of the team games will be reported in the future.

As the scheduling algorithm, we will attempt to use the search algorithm employed in strategic games such as Chess or Shogi [9]-[12]. However, in the presented game model, players move simultaneously and the number of players can be more than two. Hence, the search algorithm in its conventional form cannot be applied to this game model.

In this paper, we present a novel search algorithm suitable for the game model in which a move toward an object along the shortest path is taken as the basic move, and the move is stopped halfway when an opponent has the advantage. We refer to this algorithm as the simultaneous move search algorithm. In the algorithm, the moves of players can be dealt with quasi-alternately. However, a player can make several consecutive moves and a move already made toward an object may be aborted.

A position evaluation method based on the concept of territory is also presented. Based on the algorithm and the position evaluation method, a simulation program to play the game is developed and competitions are held among computers or between a computer and a human being. It is confirmed that the algorithm and the position evaluation method can be effectively applied to the game model.

II. Game Model of the Competition

We will take the game model as follows;

[Rule 1] The number of players is P. Players are expressed as player #1, player #2, …, and player #P. The players are also expressed as player A, player B, player C, ….

[Rule 2] Players move around in a given area which is referred to as the “field”. Players can move along any path in the area.

[Rule 3] Several objects are distributed in the field. The number of objects is a variable. Each object has a value expressed by points. An object can occupy a division which is

referred to a unit division. The distribution of objects is determined using a random number generator.
[Rule 4] When a player reaches the location of an object, the points of the object are added to the score of the player and the object is deleted from the field.
[Rule 5] When two players attempt to reach the location of an object simultaneously, one has the advantage. Player #x has the advantage against player #y if $x<y$.
[Rule 6] All the information about objects and players can be known by each player at any time, that is, the game is a complete-information game.
[Rule 7] The winner of a game is the player with the most score.
[Rule 8] A match is composed of several games. The winner of a match is the player with the most games.

The size of the field, the number of objects, the combination of the points of objects, the number of players, and the initial position of players are arbitrary.

In Fig. 1, the initial position in the case of a field of 16x16 unit divisions, 4 players and 21 objects is shown. Symbols A, B, C and D represent players and a figure x represents an object with x points.

Concerning the game model, the theoretical value of the computational complexity is O(n!) where n is the number of objects. However the practical value of the computational complexity will be $O(C^n)$ where C denotes the average number of target objects around a player and will be 10~20.

We will define the terms used in this paper as follows;
[Definition 1] Move: The action of moving toward an object along the shortest path is referred to as a "move".
[Definition 2] Half-move: The action of stopping a move halfway toward an object is referred to as a "half-move".
[Definition 3] Territory: The area in which the distance between a player and any location in the area is smaller than a constant, R, is referred to as the territory of the player.

III. Scheduling Algorithm

For the scheduling algorithm, we will attempt to use the search algorithm employed in strategic games as Chess or Shogi [9]-[12]. However, in the proposed game model, players move simultaneously, that is, not alternately, and the number of players can be more than two. Hence, the search algorithm in its conventional form cannot be applied to this game model.

A. *Dealing with Simultaneous Moves*

We present a novel search algorithm capable of dealing with simultaneous moves in which a motion to go toward an object along the shortest path is taken as the basic move and the move is changed to a half-move when opponents have the advantage.

In the algorithm, it is basically possible to deal with the actions of players as alternate actions, however there are two problems. (1) Players do not act alternately at any time. (2) The move already made by a player must be cancelled when an opponent is going to reach the same object earlier.

Problem (1) can be solved by assigning the turn to the player whose moving time is least.

Problem (2) can be solved without any contradictions by introducing the half-move, that is, when there is a possibility that a move may be cancelled, the player stops the move at the location where cancellation never happens.

B. *Dealing with Mulitple Players*

The number of players can be more than two in this game model. This problem can be solved by replacing the conventional MIN-MAX theory to the MAX-MAX theory as follows;

In the conventional MIN-MAX theory, the first player selects the move with the maximum value under the assumption that the second player will select the move with the minimum value from the view point of the first player.

In the MAX-MAX theory for two players, the first player selects the move with the maximum value under the assumption that the second player will select the move with the maximum value from own view point. Similarly, in the MAX-MAX theory for P players, the x-th player selects the move with the maximum value under the assumption that the (x+1)-th player will select the move with the maximum value from own view point.

C. *Simultaneous Moves Search Algorithm*

The algorithm is written using variables below.
C[x] (x=1~P): moving time of each player
T: turn, that is, T represents the present player.
L: search level, in other words, search depth
L_{max}: maximum value of L
M: move
V[x] (x=1~P): position evaluation value of each player
V_{min}: minimum value of V
M_b: temporary better move obtained at a given search level
V_b[x] (x=1~P): position evaluation value of each player for M_b
M_{best}: best move at a given search level
V_{best}: position evaluation value for M_{best}
Q: number of remaining objects
N: number of search members, in other words, number of moves in the search list
N_{max}: maximum number of search members

The algorithm to obtain only a single move at a given position, that is, the best move at the position will be shown here. The algorithm is written in a recursive sub-routine form as Algorithm1. The information about the present position, that is, the remaining objects, the location of each player C[x] (1~P) and L=0 is given from the main routine.

[Algorithm 1] Simultaneous move search algorithm
(Step 1) If $L=L_{max}$ or Q=0, V[x] (x=1~P)=position evaluation value of each player and return.
(Step 2) <u>T=y, C[y] is the minimum of C[x] (x=1~P).</u>
(Step 3) List possible moves. All the moves toward remaining objects are possible moves.

The list is sorted in the order of distance between the objects and the player. If N> N_{max}, then only N_{max} objects are selected according to the order.
If an opponent has already started moving toward the same object and has the advantage, the move must be deleted.
If any opponents have not yet started moving toward the same object but have the advantage, the move is changed to a half-move.
(Step 4) If the search member is the first on the list, the followings are performed.
M_b=the first member of the list.
$V_b[x]$ (x=1~P)=V_{min}.
(Step 5) Pick up a search member from the list.
M=the search member. Execute M.
The points of the object are added to the score of the player and the object is deleted from the field.
When M is a half-move, neither the addition of the points nor the deletion of the object is performed.
(Step 6) C[T]=C[T]+(time required to reach the object).
(Step 7) L=L+1. Call this subroutine recursively.
(Step 8) L=L-1. Restore the move, the score and the deleted object.
(Step 9) If V[T]>V_b[T], V_b[x]=V[x] (x=1~P) and M_b=M.
(Step 10) If there are any search members in the list, go to Step 5.
(Step 11) V[x]= V_b[x] (x=1~P) and M_{best}=M_b.
(Step 12) Return.
(End)

In the Algorithm 1, L, L_{max}, V_{min} and N_{max} are global variables, and all the other variables are used recursively, that is, the same name denotes a different variable if the level of the search, L is different.

The underlined descriptions are different from those of the conventional search algorithm [9],[10].
(1) In Step 2, the turn is assigned to the player whose moving time is least.
(2) In Step 3, the move is changed to a half-move when opponents have the advantage.
(3) In Step 5, a half-move affects nothing for the object.
(4) In Step 8, the MAX-MAX theory is realized.
(5) In Step 11, V[x]= V_b[x] realizes the renewal of V_{best} which corresponds to V_{best}=V_b in the conventional algorithm.

For the time used to stop the move halfway, we will take the time when the opponent arrives at the object.

D. Simultaneous Move Search Algorithm with Alpa-Beta Like Pruning

In the case for two players, a pruning similar to alpha-beta pruning in the conventional search algorithm [9],[10] can be introduced to the presented search algorithm, although continuous moves have to be take into account. The simultaneous move search algorithm with pruning is written as Algorithm 2. The initial values of alpha and beta, that is, alpha=V_{min}, beta=V_{min} are given from the main routine.

[Algorithm 2] Simultaneous move search algorithm with alpha-beta like pruning

(Step 1) If L=L_{max} or Q=0, V=position evaluation value and return.
(Step 2) T=y, C[y] is the minimum of C[x] (x=1~P).
(Step 3) List possible moves. All the moves toward remaining objects are possible moves.
The list is sorted in the order of distance between the objects and the player. If N> N_{max}, then only N_{max} objects are selected according to the order.
If an opponent has already started moving toward the same object and has the advantage, the move must be deleted.
If any opponents have not yet started moving toward the same object but have the advantage, the move is changed to a half-move.
(Step 4) If the search member is the first on the list, the followings are performed.
M_b=the first member of the list.
$V_b[x]$ (x=1~P)=V_{min}.
(Step 5) Pick up a search member from the list.
M=the search member. Execute M.
The points of the object are added to the score of the player and the object is deleted from the field.
When M is a half-move, neither the addition of the points nor the deletion of the object is performed.
(Step 6) C[T]=C[T]+(time required to reach the object).
(Step 7) L=L+1. Call this subroutine recursively.
(Step 8) L=L-1. Restore the move, the score and the deleted object.
(Step 9-1) If V[T]>V_b[T], V_b[x]=V[x] (x=1~P) and M_b=M.
(Step 9-2) If (T=0 and V_b[T]>beta) or (T=1 and V_b[T]>alpha), then pruning, that is, stop the repeating and go to Step 12, else go to Step 9-3.
(Step 9-3) If T=1, then alpha= V_b[T], else beta= V_b[T].
(Step 10) If there are any search members in the list, go to Step 5.
(Step 11) V[x]= V_b[x] (x=1~P) and M_{best}=M_b.
(Step 12) Return.
(End)

The underlined descriptions are different from those of the Algorithm 1. By applying Algorithm 2, the maximum search level is greatly increased, that is, almost twice that of Algorithm 1, however Algorithm 2 can be applied only for the case including two players.

IV. Position Evaluation

To realize the presented search algorithm, the position evaluation value at a given position has to be calculated independently on every player and has to be a positive value.

In this game model, the following factors will contribute the position evaluation.
(1) the score already obtained by player #x, S[x]
(2) the future score obtainable by player #x, F[x]
(3) the scores already obtained by the players other than #x, S[y]
(4) the future scores obtainable by the players other than #x, F[y]

We present the position evaluation as follows, where V[x] is the evaluation value of player #x.

$V[x]=S[x]+F[x]-(S[y]+F[y])$ (1)

$F[x]=G[x]K$ (2)

$F[y]=G[y]K$ (3)

where G[x], G[y] and K denote the total points of the objects in the territory of player #x, the total points of the objects in the territory of the players other than #x and a coefficient, respectively. We refer to K as the position evaluation coefficient which is empirically determined by carrying out a large number of matches.

The territory defined by "Definition 3" becomes a circle with radius R, however a recognition that an object is inside a circle or not, requires much calculation time, hence the territory circle will be approximated by a square with 2Rx2R area. The appropriate value of R depends on the size of field, the number of objects, etc. We will take R=(the area of the field)/8 here.

V. Simulation Program and Results of Games

A simulation program is developed and competitions are held among computers.

Figure 2 shows the running state of the game with two players for the initial position illustrated in Fig. 1. Only player A and player B are attending to the game. Triangles represent players. For example, the triangle at the upper position is player A. A square represents an object and the size of each square is proportional to the points of the object. The lines between the objects represent the tracks of the players.

In Fig. 2, the search parameters are determined to L_{max}=1, and N_{max}=1 which imply that the nearest object is selected as the target. From Fig. 2, we see that the program is correctly running..

Figure 3 shows the running state of the game with 4 players for the same initial position as Fig. 2.

Figure 4 is an example of the running state of a game with 100 objects.

Following results are obtained for the case of two players.

(1) The handicap between the player with the advantage and the other is negligible.

(2) The maximum number of search members, N_{max}=8 is enough.

(3) The maximum search level, L_{max} is 6 for without pruning and is 10 for with the alpha-beta pruning, where objects are 41 and the calculation time is limited to 2 seconds.

(4) The position evaluation using K is effective and the optimum value of K is 0.5~1 as shown in Fig. 5.

In Fig. 5, L_{max}=6, N_{max}=8, the number of games is 1000, the win is expressed from the view point of player A, n denotes the number of objects and pt denotes the pattern of the objects, where pt=1 represents all the objects have 1 point, and pt=3 represents that one object has 3 points, 3 objects have 2 points and the others have 1 point.

VI. Evaluations

For the competing algorithms in evaluation, we use following two.

(1) the greedy method[13]

(2) the heuristic algorithm.

In the greedy method, the nearest object is selected as the target. It is simple method but can be effectively employed to obtain an approximate solution for scheduling. The greedy method is realized by setting L_{max}=1 and N_{max} =1 for a player in the developed program.

Figure 6 shows the results of games for two players in which L_{max} is varying only for player A and fixing L_{max}=1 for player B, that is, player A is the presented method and player B is the greedy method here. Other conditions are N_{max} =8, K=1, the number of games is 1000, and the win is expressed from the view point of player A.

From Fig. 6, we see that the winning average of the presented method greatly increases along with the value of L_{max} and becomes 99% when L_{max} =12. Hence we can state that the presented algorithm is effectively applied to the competition here.

Table I shows the results of games between the presented method and the heuristic algorithm. Heuristic algorithm here implies university students who can see the present position on the display panel and will search the target object considering the distance to the object, the cluster of objects, and the advantage against the opponent, etc.

In Table I, the number of objects, n is 21 and the pattern of the objects, pt is 3, L_{max} =12, N_{max} =8, the number of games is 20 for each player, and the win is expressed from the view point of the presented method.

From Table I, we see that the presented method under the strongest conditions, that is, L_{max} =12 has the winning average of 50~80% against human beings.

The reason why the presented method is superior than the students will be that the presented method has deeper search level. In Table I, the results of the greedy method vs. the students are also shown. From Table I, the students have the winning average of 60~80% against the greedy method. By applying this average to Fig. 6, we see that the students here have the ability corresponding to the presented method with L_{max} =4.

Figure 7 shows the results of games for two players, three players and four players. In Fig. 7, the conditions are similar to those in Fig. 6, that is, player A is the presented method, other players are the greedy method, N_{max} =8, n=21, and pt=3.

From Fig. 7, we see that the winning average of the presented method is increased along with the value of L_{max}, however the increasing rate becomes slower along with the number of players, P. The reason why the increasing rate becomes slower will be that taking leadership of a game becomes difficult when the number of players is increased.

VII. Conclusions

In this paper, we have studied a traveling-salesman type competition in which several players travel around various objects distributed in a given area and compete with each other in collecting the objects.

A search algorithm suitable for the competition is presented in which a move toward an object along the shortest path is taken as the basic move and the move is stopped halfway when opponents have the advantage. A position evaluation method based on the concept of territory is also presented. A simulation program is developed and competitions are held among computers or between a computer and a human being. It is confirmed that the algorithm and the position evaluation can be effectively applied to the competition.

Following tactics will be useful for the competition and will be studied in the future. (1) go straight toward the cluster of objects, (2) go toward the objects apart from the cluster of opponents, (3) partial application of the conventional traveling-salesman-problem algorithm, that is, the algorithm without competitor.

The team games in which the member of a team cooperate with each other and compete against the opposing team will be also studied in the future.

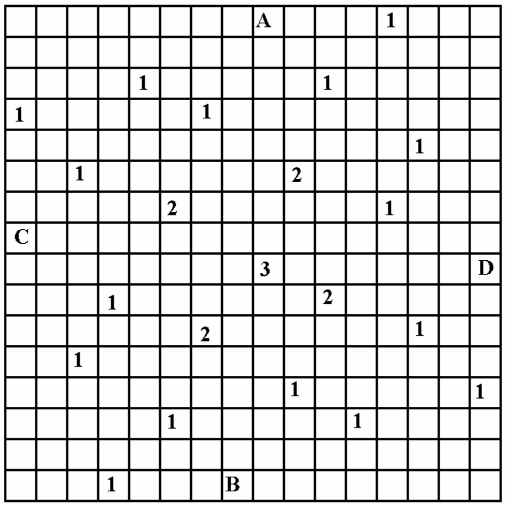

Fig. 1 An example of the initial position

Symbols A, B, C and D represent players.
A figure x represents an object with x points.

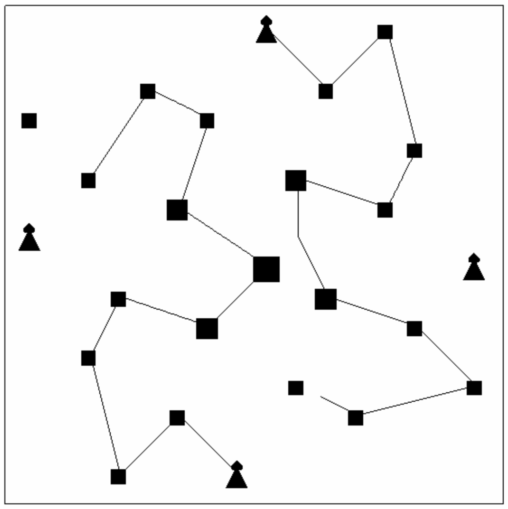

Fig. 2 Running state of the game with 2 players
Triangles represent players. A square represents an object and the size of each square is proportional to the points of the object. The lines between the objects are the track of the players.

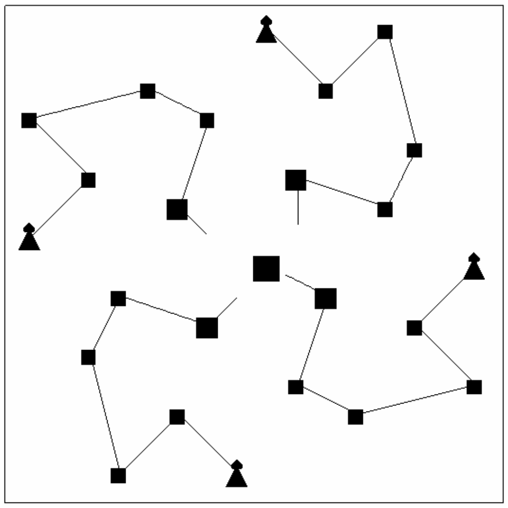

Fig. 3 Running state of the game with 4 player

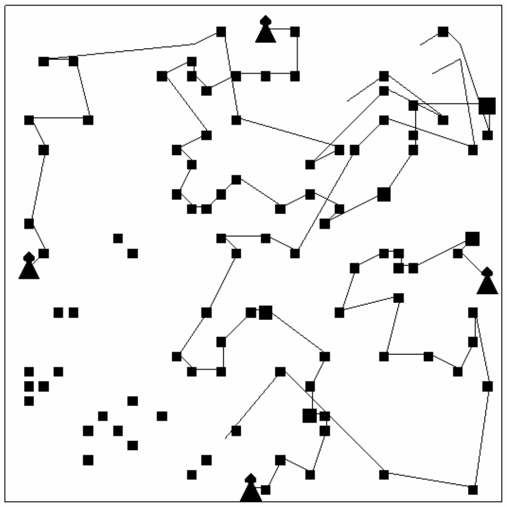

Fig. 4 Running state of the game with 100 objects

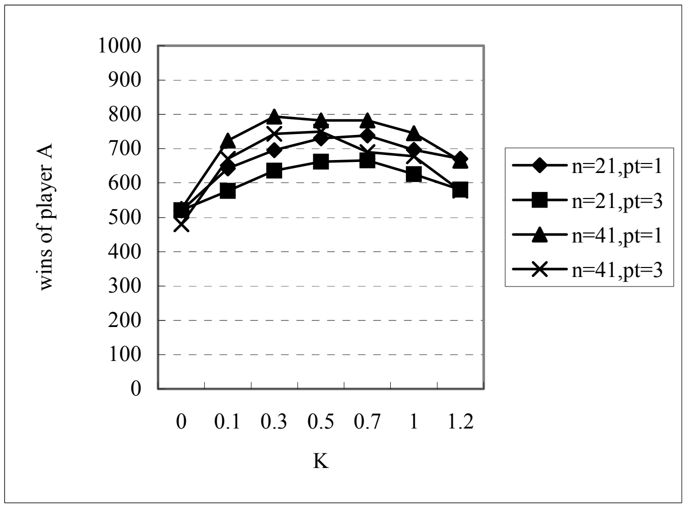

Fig. 5 Results of games for different values of K
Only K of player A is varied, K=0 for player B

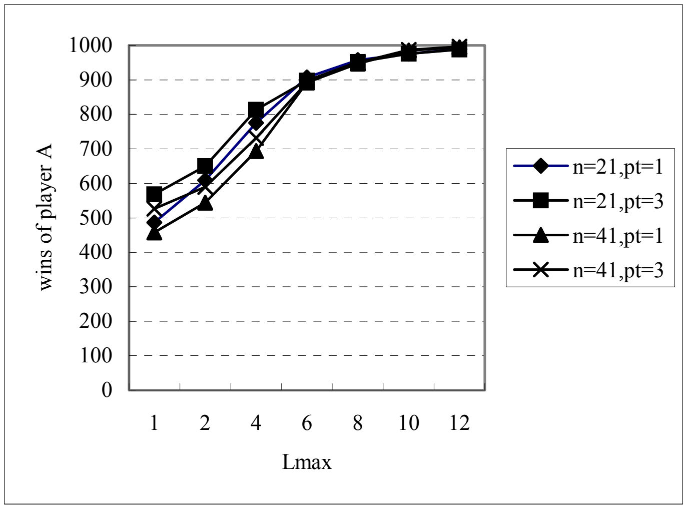

Fig. 6 Results for proposed method vs. greedy method
player A: proposed method, player B: greedy method

TABLE I. Results for proposed method vs. human being

		Proposed method (wins - losses)	Greedy method (wins - losses)
opponent	Student A	16 - 4	8 - 12
	Student B	15 - 5	6 - 14
	Student C	13 - 7	8 - 12
	Student D	15 - 5	7 - 13
	Student E	10 - 10	4 - 16

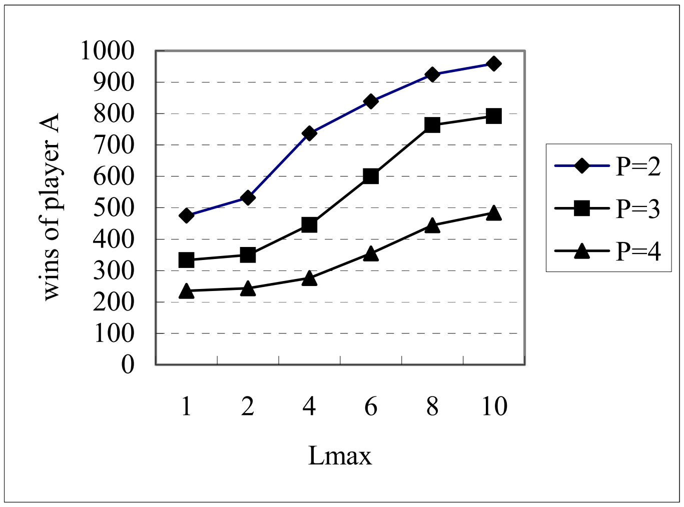

Fig. 7 Results for proposed method vs. greedy method
The number of players, P is varying.

REFERENCES

[1] G. Reinelt, The Traveling Salesman –Computational Solutions for TSP Applications, Springer-Verlag, 1994.

[2] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook, The Traveling Salesman Problem, Princeton University Press, 2006.

[3] N. Tomabechi, T. Fujieda, and Y. Fujioka, "Design of a search processor for competitively traveling robots," Proc. of Meditterranean Conference on Control and Automation, MoM1-A5, pp. 1-6, June 2004.

[4] N. Tomabechi and Y. Fujioka, "Traveling-salesman type competition and simultaneous move search algorithm," Proc. of ECTI-CON 2009, pp. 682-685, May 2009.

[5] A. Namatame, S. Arai and K. Hattori, "Interactions between agents," Journal of the Society of Instrument and Control Engineers Japan, Vol. 44, No. 12, pp. 865-874, Dec. 2005.

[6] H. Akiyama et. al., "Team formation construction using a GUI tool in the RoboCup Soccer simulation, Proc. of SCIS & ISIS 2006, 2006.

[7] Y. Kobayashi and S. Hosoe, "A autonomous distributed control of multi-agent systems," Journal of the Society of Instrument and Control Engineers Japan, Vol. 46, No. 3, pp. 178-184, March 2007.

[8] S. Okada, Y. Ito, and O. Hasegawa, "Generation of adapting behavior by multi-robot using parameter searching on subsumption architecture," Transactions of the Japanese Society for Artificial Intelligence, pp. 276-290, Vol. 22, No. 3, D, 2007.

[9] D. Levy and M. Newborn, How Computer Play Chess, W. H. Freeman and Company, New York, 1990.

[10] Y. Kotani, T. Yoshikawa, Y. Kakinoki, and K. Morita, Computer Shogi, Science Company, Tokyo, 1990.

[11] E .A. Heinz, Scalable Search in Computer Chess, Vieweg, 2000.

[12] I. Ishikawa, K. Matsubara, J. Nagashima, Y. Kajihara, T. Hashimoto, and H. Iida, "A self-play experiment in Computer Shogi," Proc. of the 9-th Game Programming Workshop, pp.40-47, Nov. 2004

[13] J. Kleinberg and E. Tardos, Algorithm Design, Addison-Wesley, 2005.