Journal of the Operational Research Society (2013) 64, 208–216

www.palgrave-journals.com/jors/

Competitive travelling salesmen problem: A hyper-heuristic approach

G Kendall[1] and J Li[2*]

[1]*The University of Nottingham Malaysia Campus, Selangor, Malaysia; and* [2]*The University of Nottingham, Nottingham, UK*

We introduce a novel variant of the travelling salesmen problem and propose a hyper-heuristic methodology in order to solve it. In a competitive travelling salesmen problem (CTSP), *m* travelling salesmen are to visit *n* cities and the relationship between the travelling salesmen is non-cooperative. The salesmen will receive a payoff if they are the first one to visit a city and they pay a cost for any distance travelled. The objective of each salesman is to visit as many unvisited cities as possible, with a minimum travelling distance. Due to the competitive element, each salesman needs to consider the tours of other salesman when planning their own tour. Since equilibrium analysis is difficult in the CTSP, a hyper-heuristic methodology is developed. The model assumes that each agent adopts a heuristic (or set of heuristics) to choose their moves (or tour) and each agent knows that the moves/tours of all agents are not necessarily optimal. The hyper-heuristic consists of a number of low-level heuristics, each of which can be used to create a move/tour given the heuristics of the other agents, together with a high-level heuristic that is used to select from the low-level heuristics at each decision point. Several computational examples are given to illustrate the effectiveness of the proposed approach.

Journal of the Operational Research Society (2013) **64,** 208–216. doi:10.1057/jors.2012.37;
published online 25 April 2012

Keywords: travelling salesmen problem; heuristics; game theory; hyper-heuristic

1. Introduction

The travelling salesmen problem (TSP) is one of the most intensively studied problems in operational research, and even beyond. Given a number of cities and the distances between them, the task is to find the shortest tour that visits each city exactly once, returning to the starting city (ie, a Hamiltonian cycle). Many variants of TSP have been developed, for example, multiple TSP, asymmetric TSP, TSP with time windows and so on. In fact, the TSP has been so intensively studied that we cannot do justice to the vast number of papers that have been written, but we refer the reader to (Lawler *et al*, 1986; Gutin and Punnen, 2002; Gendreau and Potvin, 2005; Applegate *et al*, 2006; Laporte, 2010).

As a variant of the TSP, the competitive travelling salesmen problem (CTSP) is a hybridization of decision making and optimization. In a CTSP, multiple travelling salesmen compete with each other in visiting a number of cities. The salesmen (or agents, as we shall refer to them) will receive a payoff if they are the first one to visit a city and they will pay a cost for the distance that they travel. Therefore, each agent aims to visit as many unvisited cities as possible with the minimum travelling distance. It is an NP-complete problem for the agents to plan their tours and there is strategic interaction between the agents because of their conflict of interest.

The difference between CTSP and the multiple TSP is that the multiple TSP can be treated as a cooperative game, in which multiple agents have an identical objective and they collaborate in order to minimize the aggregated travel distance. The CTSP is a non-cooperative game in which multiple agents are selfish and they only care for their own payoff. A number of approaches for multiple TSP have been developed (Bektas, 2006). However, most of those approaches cannot be used to solve CTSP directly.

A real-world instance of the CTSP could be viewed as a set of travelling circuses. In former times, most circuses needed to travel from town to town to find large enough audiences. When planning their tours, they should consider the distance travelled because increased travel reduces their profit. On the other hand, they need to consider the tours of other circuses. They should avoid travelling to the cities recently visited by other circuses as people will be less

*Correspondence: J Li, School of Computer Science, The University of Nottingham, Jubilee Campus, Wollaton Road, Nottingham, UK NG8 1BB, UK.
E-mail: jwl@cs.nott.ac.uk

inclined to visit a different circus after having just seen another one.

Advertising is another example. Consider the scenario in which several enterprises compete in a number of markets over a long period of time. Each enterprise needs to determine how to assign their limited budget and what means of advertising to adopt in different markets in order to maximize profits. If we denote the combinations of markets and means of advertising by a set of cities, it becomes a CTSP where the distance to each city is the expenditure.

A CTSP is a complex non-cooperative game. On the basis of the concept of Nash equilibrium, game theorists have developed the theory of equilibrium analysis for non-cooperative games (Nash, 1951; Osborne and Rubinstein, 1999; Papadimitriou and Roughgarden, 2005). The theoretical solution to CTSP is Nash equilibrium that denotes the state in which no agent can be better off by unilaterally changing their strategy. The Nash equilibrium is a stable state because the agents have no motivation to deviate from the state once it has been implemented.

However, equilibrium analysis can only be applied to very simple CTSPs and it is too complex to compute the Nash equilibrium(s) when the number of cities is large. Even if all the tours of other agents are given, it is still an NP-hard problem for one agent to compute the optimal tour. No algorithm (unless $P = NP$) guarantees the optimal solution can be found in polynomial time. Therefore, the agents cannot be sure what their optimal strategy should be and CTSP becomes a game with uncertainty. Second, CTSP is a dynamic game. The agents need to consider not only the sequences of cities to visit but also when to visit those cities. So time is also a parameter of their strategies, which makes the problem even more complicated.

In this study, we investigate a hyper-heuristic approach for CTSP, as equilibrium analysis and other approaches that have been used for TSP cannot be directly utilized. A hyper-heuristic can be seen as a high-level heuristic, which generates a solution to search problems by selecting, combining, generating, or adapting a given set of simple heuristics. Hyper-heuristics are considered to be methodologies to build systems, which can handle different classes of search problems. They have been applied to personnel scheduling (Cowling *et al*, 2000), timetabling (Burke *et al*, 2003a, 2005), space allocation (Bai *et al*, 2008), packing problems (Burke *et al*, 2006, 2010a), and vehicle routing problems (Pisinger and Ropke, 2007; Garrido and Riff, 2007). Introductory articles on hyper-heuristics can be found in Burke *et al* (2003b), Ross (2005), Burke *et al* (2009), and Burke *et al* (2010b).

This article is structured as follows. Section 2 defines the CTSP and discusses heuristic approaches as a way of solving it. Section 3 presents the proposed hyper-heuristic for the CTSP. The results of two computational examples are given in Section 4. Finally, Section 5 discusses our results and suggests directions for future work.

2. Competitive travelling salesman problem

As opposed to the multiple TSP, the competitive TSP is a dynamic game with strategic interaction of the behaviours of multiple players. Let $M = \{1, \ldots, m\}$ denote the set of agents. They compete with each other in visiting n cities. The agents are self-interested, only taking into consideration their own payoffs. Therefore, the game is non-cooperative. We define the following conditions for CTSP.

- *Benefit*. For each city $i \in \{1, \ldots, n\}$, there is a constant benefit B_i for the agent who is the first one to reach the city. The other agents receive zero benefit if they subsequently visit the city. If two or more agents arrive at an unvisited city simultaneously, they will share the benefit equally.
- *Cost*. Each agent has to pay a cost for their travel that is proportional to their travel distance. Let C_{ij} denote the cost for an agent travelling from City i to j.
- *Payoff*. The payoff for each agent is computed by aggregating the benefit and cost. Assume that an agent visits k cities and receives benefit from k_0 ($k_0 \leqslant k$) cities. The payoff received by the agent can be computed as

$$u = \sum_{j=1}^{k_0} B_j - \sum_{i=1}^{k-1} C_{i(i+1)} - C_{k1}$$

  The $k - k_0$ visits are *wasted* visits. A wasted visit here denotes that an agent travels to a city that has been visited by another agent(s) and thus receives no benefit. For simplicity, we use $\sum_{j=1}^{k_0} B_j$ to express the aggregated benefit the agent receives from k_0 cities.
- *Path*. The agents are initially located in different cities. The agents must travel from one city to another and they cannot change their destinations once they have started a trip. They must return to their departure city to complete their travel.
- *Speed of travel*. Each agent $k \in \{1, \ldots, m\}$ has a constant speed of travel v_k.
- *Common knowledge*. The location, the speed of travel, the path travelled, and the payoff for each agent are known to all agents.

Under these conditions, each agent chooses their tour independently. The objective of each agent is to maximize their own payoff. In other words, each agent aims to visit as many unvisited cities before any other agents, while also minimizing their distance.

Let s_k, $k \in \{1, \ldots, m\}$ denote the strategy of Agent k, S_k the set of strategies of Agent k. Let s_{-k} denote the profile of

strategies of all agents except k, S_{-k} the set of strategies of all agents except k. Each strategy s_k leads to a specific tour and corresponding payoff for Agent k given s_{-k}. If a strategy vector $s \in S$ is a Nash equilibrium, then for all Agents k and each alternate strategy $s'_k \in S_k$, there must be

$$u_k(s_k, s_{-k}) \geqslant u_k(s'_k, s_{-k}) \quad (1)$$

where $u_k(s_{k,} s_{-k})$ denotes the payoff of Agent k, given the strategy profile of $(s_{k,} s_{-k})$.

Since CTSP is a dynamic game, a strategy consists of a sequence of choices made by an agent during the game. In order to describe the strategies in CTSP, we define a *move* as a single choice made by an agent.

An individual agent has at most n non-stochastic alternatives (cities) to choose at each move (a move can also be a 'mixed strategy' as defined in game theory, that is, two or more cities are chosen with a distribution of probabilities). In some circumstances, an agent may need to travel to a city that has been visited by another agent. That is, they have to make a *wasted visit*. Consider an example of a two-agent, six-city CTSP as shown in Figure 1. Assume that two agents have the same speed of travel and the payoff received by visiting a city is equal for all cities. We assume that the benefit received by visiting a city is higher than the maximum cost of travel. Thus, the agents have the incentive to visit all cities. The optimal tour for Agent 1 is *ABCDA*, while the optimal tour for Agent 2 is *ECFE*, as shown in Figure 1. It is necessary for Agent 1 to reach City C (that has been visited by Agent 2) before moving to D. If Agent 1 chooses the tour of *ABDA*, Agent 2 can be better off by choosing the tour of *ECDFE* and Agent 1 will make a wasted visit to City D. This is another difference between CTSP and the multiple TSP. Wasted visits are not allowed in the latter version.

In some circumstances, an agent may need to stay in a city waiting for other agents to make their decisions. Consider the following example of a two-agent, six-city CTSP. Assume that two agents have the same speed of travel and the payoff received by visiting a city is equal for all cities. The optimal tour for Agent 1 is *ABCA*, whereas the optimal tour for Agent 2 is *DEFD*, as shown in Figure 2. Agent 2 needs to wait at *E* until Agent 1 has started moving from *B* to *C*. If Agent 2 does not wait, Agent 1 can be better off by changing the tour to *ABFCA* and Agent 2 will make a wasted visit to City *F*. The agents' travel distance is a function of time in the Nash equilibrium as shown in Figure 3. Changing either of the sequence of cities or time to visit them will deviate from the Nash equilibrium.

Theoretically, the solution to a CTSP is a Nash equilibrium(s) that denotes the state in which each agent cannot improve their payoffs by unilaterally changing their strategy. However, it is too complex to find the Nash equilibrium when the number of cities is large. The Nash equilibrium has an effect on the agents in choosing their strategies only if it is exactly known by all the agents involved. If an agent does not know the Nash equilibrium, or they believe that some others do not know it, a Nash equilibrium is no longer a stable solution. Because of the difficulty in applying equilibrium analysis, we introduce a hyper-heuristic methodology for the CTSP.

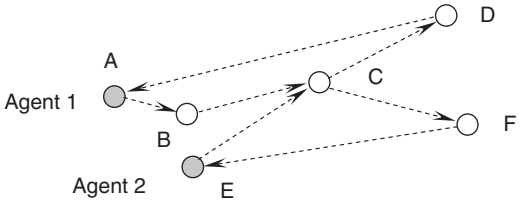

Figure 1 A two-agent CTSP. Agent 1 has to make a wasted visit to City *C*. Agents 1 and 2 are initially located at Cities *A* and *E*, respectively.

3. A hyper-heuristic model for CTSP

Our model assumes that each agent in a CTSP adopts a set of heuristics to perform their moves and thus create a tour. Since a heuristic does not guarantee the optimal solution, each agent knows that his/her move/tour, as well as other agent's moves/tours, is not necessarily optimal.

We assume that the agents have a limited set of heuristics available, H. The sets of strategies, S_k and S_{-k}, are then replaced by the sets of heuristics, H_k and H_{-k}. The payoff of Agent k is denoted by $u_k(h_k, h_{-k})$, where h_k is the heuristic adopted by Agent k and h_{-k} the profile of heuristics adopted by all agents except k. The agents choose between different heuristics instead of strategies, according to the corresponding payoff matrix. For example, consider a two-agent CTSP. The payoff matrix can be expressed as shown in Table 1, where a limited number of l heuristics are taken into

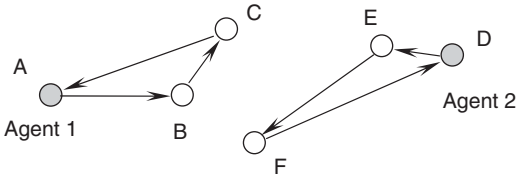

Figure 2 An example of two-agent six-city CTSP. Agents 1 and 2 are initially located at Cities *A* and *D*, respectively.

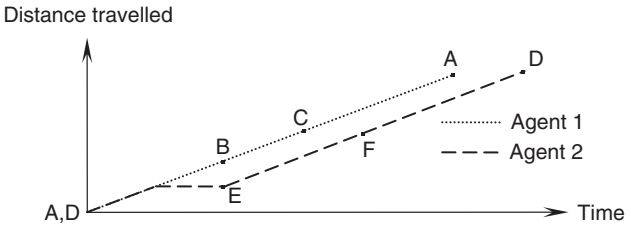

Figure 3 The travel distances as functions of time in Nash equilibrium.

Table 1 The payoff matrix for the two-agent CTSP

Agent A	*Agent B*			
	h_1	h_2	...	h_1
h_1	$u_A(h_1, h_1)$, $u_B(h_1, h_1)$	$u_A(h_1, h_2)$, $u_B(h_1, h_2)$	...	$u_A(h_1, h_1)$, $u_B(h_1, h_1)$
h_2	$u_A(h_2, h_1)$, $u_B(h_2, h_1)$	$u_A(h_2, h_2)$, $u_B(h_2, h_2)$	...	$u_A(h_2, h_1)$, $u_B(h_2, h_1)$
...	...	...	...	...
h_1	$u_A(h_1, h_1)$, $u_B(h_1, h_1)$	$u_A(h_1, h_2)$, $u_B(h_1, h_2)$	...	$u_A(h_1, h_1)$, $u_B(h_1, h_1)$

consideration. u_A and u_B are the payoffs of Agents A and B, respectively. Each agent will choose heuristics in order to maximize their own payoff.

Now the problem turns out to be computing the equilibrium (a profile of heuristics of all agents) of a static game. The payoff matrix has m dimensions and l^m items for an m-agent CTSP, assuming that each agent has l heuristics.

It is potentially difficult to compute the equilibria because there is no algorithm with less than exponential computational complexity to compute the equilibria of the games with two or more players. If we concentrate on pure heuristics and ignore 'mixed heuristics', the computation will be greatly simplified. Here, mixed heuristic denotes the agents' choice that assigns probabilities over different pure heuristics.

Since CTSP is a dynamic game, there are two ways for the agents to make their choices. An agent can either choose a heuristic at the beginning and then apply it for the whole game, or choose a different heuristic on different moves. In the latter case, the agent actually adopts a hyper-heuristic to choose a combination of different heuristics.

Our proposed hyper-heuristic is a combination of game theory of incomplete information and heuristics. The hyper-heuristic consists of a number of low-level heuristics and a high-level heuristic. The low-level heuristics are a set of simple heuristics, each of which can be used to create a move/tour given the heuristics of other agents. The high-level heuristic identifies the heuristics adopted by other agents and then computes, and chooses, the heuristic that leads to the highest payoff.

The low-level heuristics are constructive heuristics to create a move or tour. Although many heuristics for the TSP have been developed, most of them cannot be directly applied to CTSP. In this study, five low-level heuristics are used.

1. Nearest Neighbour (NN). This heuristic always chooses the nearest unvisited city as the next destination.
2. Random Neighbour (RN). This heuristic randomly chooses one of the 'neighbour' cities as the next destination. Those cities are defined as neighbour cities where the distance is not $>120\%$ of the nearest one.
3. Aggressive (AH). This heuristic aims to avoid wasted visits and also increase other agents' wasted visits. It first checks other agent's destination and it chooses another's destination if it takes a shorter time to reach the city than other agents. If not, it chooses the nearest unvisited city if no other agents can visit it first. If not, it checks the second nearest, the third nearest ..., until it finds a city that no other agents can reach first. If no city can be found, it waits for one time step at the current location.
4. NN + 2-opt. This heuristic first adopts NN to create a tour, and then does a 2-opt local search to improve it. It assumes that other agents use the NN heuristic (Note, 2-opt repeatedly swaps any two cities of a tour to find a better tour with smaller cost).
5. RN + 2-opt. This heuristic is the same as NN + 2-opt except that RN is adopted to construct the tour.

Heuristics 3–5 take into consideration the heuristics of other agents and Heuristics 4 and 5 involves a local search. The search considers at most 30 cities so as not to increase the computational time to unacceptable limits.

A low-level heuristic is adopted to choose the target city to visit on each move. Although some low-level heuristics, for example, Heuristics 4 and 5, create tours, only the first city in a tour will definitely be visited because the high-level heuristic computes the expected payoffs and chooses among different heuristics at every move.

The high-level heuristic identifies the heuristics adopted by the other agents according to their historical moves, and then selects the heuristic with the highest expected payoff to deal with them. The evaluation of the heuristics adopted by other agents are expressed by *belief*s. A belief is a 5-tuple, $\{P_1, P_2, P_3, P_4, P_5\}$, which denotes the probability that a specific low-level heuristic is adopted. For example, at the beginning of a CTSP, Agent i has a belief of $\{0.2, 0.2, 0.2, 0.2, 0.2\}$ about another Agent j's heuristic, meaning that i believes that j is likely to adopt any of the heuristics with the same probability.

The latest three choices of other agents are used to compute the beliefs at each move, and the heuristic with the highest probabilities will be chosen as the other agents' heuristic. For example, if Agent j has chosen another agent's destination in the latest three choices, i's belief about j will be $\{0, 0, 1.0, 0, 0\}$ and j will be assumed to adopt AH for the following move. Bayes theorem can be used to compute the beliefs given the agents previous choices.

$$p(A|B) = \frac{p(B|A)p(A)}{p(B)} \tag{2}$$

where A and B are two correlated events. For example, if A is *the agent adopts NN* and B is *the agent chooses city* c_i, $p(A|B)$ gives the posterior probability of A given B.

In order to reduce the computational effort, we adopt a look-up table to compute the beliefs. The look-up table is a group of if-then rules and it is generated by using both reasoning and Bayes theorem. It follows the following rules:

If the agent chooses the nearest cities three times, then the heuristic adopted is NN.

If the agent chooses another agent's destination, then the heuristic adopted is AH.

Given the heuristics of other agents, an imaginary tour is created by adopting each low-level heuristic so that the payoff can be computed. The hyper-heuristic will then choose the heuristic with the highest payoff to create a real tour for the agent. We note that only the first destination of a tour will definitely be visited because the beliefs and tour are computed and updated after every move. In order to limit the amount of computation, each imaginary tour contains at most 30 moves (depending on how many unvisited cities are left). In pseudo-code, the hyper-heuristic is as follows:

1. Check whether there is an unvisited city. If not, go to 6.
2. Compute beliefs according to Bayes Theorem, see Equation (2).
3. Compute the payoff of adopting each low-level heuristic given the heuristics of other agents.
4. Select the heuristic with the maximum payoff to generate the next destination.
5. Go to 1.
6. End.

This hyper-heuristic model greatly simplifies CTSPs so that some simple heuristics can be utilized to obtain approximate solutions to CTSP.

4. Computational examples

In this section, two examples are studied: two-agent CTSP with 20 cities and three-agent CTSP with 300 cities. A simple CTSP is analysed in the first example in order to show Nash equilibrium and compare it with the optimal solution of the multiple TSP.

4.1 Two-agent CTSP with 20 cities

The cities are located in a square area of 100X100 and their locations are randomly created. The payoff of visiting each city is set to be 150, while the cost of travel is equivalent to the distance travelled. Note that the payoff of visiting a city is greater than the longest distance between two cities, so the agents are motivated to visit all cities. Two agents with identical speed of travel are present and they are initially located in different cities.

We computed the Nash equilibrium of this game (see Figure 4). We note that the direction of travel (indicated by the arrows) is important in CTSP. For example, if one agent adopts the reverse direction, their tours are no longer Nash equilibrium because another agent can improve his/her payoff by changing their tour. Also, Agent 1 needs to wait at City A and cannot move to the next city until Agent 2 has started moving from City B to D. Otherwise, Agent 2 can visit 11 cities by moving to City C instead of D.

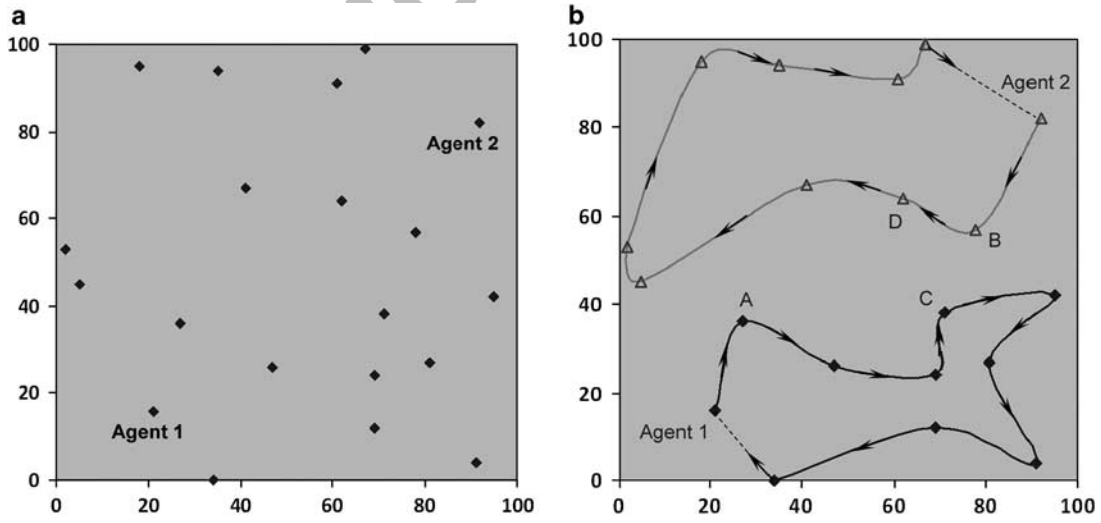

Figure 4 Nash equilibrium solution of CTSP with 20 cities. The payoff for Agent 1 is 1119.57 (distance travelled 230.43), whereas Agent 2 receives 1103.56 (distance travelled 246.44). Note that Agent 1 needs to wait at City A and cannot move to the next city until Agent 2 has started moving from City B to D. Otherwise, Agent 2 can manage to occupy 11 cities by moving to City C instead of D.

As a comparison, we compute the optimal solution of the multiple TSP that has the same map as this CTSP. Multiple TSP could be considered as a cooperative game in which multiple agents aim to maximize the social payoff. The optimal solution is shown in Figure 5(a) and the maximum social payoff is 2231.55. Figure 5(b) shows a Pareto optimal solution where two agents *fairly* share the payoff. Pareto optimal denotes the state in which each agent cannot improve their payoff without making another agent worse off.

It is much more difficult to find the Nash equilibrium for CTSP than to compute the optimal solution for a multiple TSP that has the same number of agents and cities. It is even difficult to apply an exhaustive search for CTSP because it is possible that some agents need to wait at a city for a period of time. We simulate the interaction between two agents using the heuristics of NN, RN, AH, NN + 2-opt, RN + 2-opt, and the previously described hyper-heuristic (Hyper). The values in each pair of parentheses in Table 2 are the payoffs of Agents 1 and 2, respectively. Comparing the payoffs of Hyper with other low-level heuristics, it shows that the performance of Hyper is superior in most cases, although it does not always receive the highest payoff.

In order to decrease the influence of the locations of cities, we ran an experiment in which the agents adopted different heuristics over 1000, two-agent, 20-city CTSPs. The locations of cities were randomly created in each CTSP. The results are shown in Table 3. It shows that Hyper has an obvious advantage over the low-level heuristics when used in isolation. We also added a new heuristic, Insertion + 2-opt, to investigate the performance of Hyper in dealing with unknown heuristics. This heuristic selects the nearest unvisited city and makes a subtour of it. Then it selects the unvisited city having the shortest distance to any one of the cities in the subtour, and inserts this city into the subtour such that the cost of the subtour is minimal. This process continues until the subtour contains 30 cities or no unvisited city is available. It assumes that other agents adopt NN to create their tours. Finally, 2-opt is adopted to improve the subtour. However, 2-opt does not always improve a subtour, as it does not

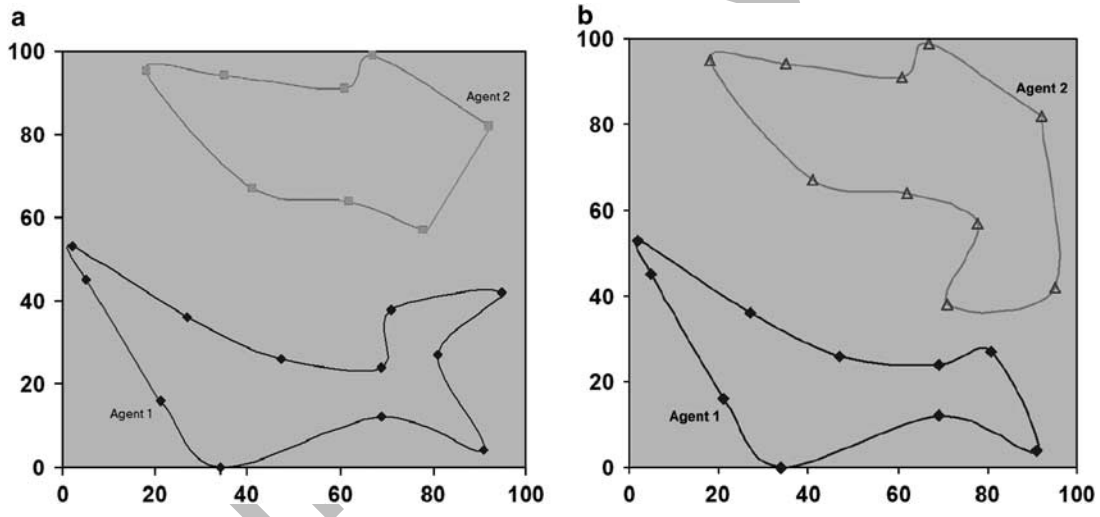

Figure 5 Cooperative solutions of a multiple TSP with 20 cities. (a) Optimal solution with the maximum social payoff. The payoff of Agent 1 is 1368.55 (travelling distance 281.45) and Agent 2 receives 863.0 (travelling distance 187.0). (b) Pareto optimal solution with which the payoff of Agent 1 is 1115.18 (travelling distance 234.82) and the payoff of Agent 2 is 1106.96 (distance travelled 243.04).

Table 2 Payoffs of adopting different heuristics (two values in each pair of brackets are the payoffs of two agents, respectively)

Agent 1	*Agent* 2					
	NN	*RN*	*AH*	*NN* + 2-*opt*	*RN* + 2-*opt*	*Hyper*
NN	(891.2, 1133.3)	(1007.8, 1029.2)	(891.2, 1133.3)	(826.7, 1169.5)	(1007.8, 1037.4)	(842.6, 1238.0)
RN	(1163.9, 873.9)	(1012.4, 1004.2)	(1289.0, 854.6)	(1140.7, 822.7)	(1010.7, 1037.4)	(1097.8, 1044.1)
AH	(948.6, 1133.3)	(1080.0, 1029.2)	(948.6, 1133.3)	(1045.5, 904.7)	(1080.0, 1037.4)	(948.6, 1185.1)
NN + 2-opt	(1354.0, 685.4)	(1072.3, 1031.4)	(1354.0, 854.6)	(1023.5, 1051.4)	(1072.3, 1062.4)	(1023.6, 1051.4)
RN + 2-opt	(1354.9, 685.4)	(1110.3, 1037.3)	(1354.0, 854.6)	(1431.3, 574.8)	(829.6, 1205.8)	(884.7, 1159.6)
Hyper	(1187.0, 856.2)	(1204.1, 826.1)	(1354.0, 854.6)	(1141.2, 895.3)	(1234.2, 790.9)	(1052.6, 1021.6)

Table 3 Average payoffs of adopting different heuristics in 1000 CTSPs

Agent 1	*Agent* 2						
	NN	*RN*	*AH*	*NN* + 2*opt*	*RN* + 2*opt*	*Hyper*	*Insert* + 2*opt*
NN	(1074.3, 1072.4)	(1086.3, 1041.1)	(1126.6, 1049.0)	(1079.2, 1063.6)	(1077.3, 1082.4)	(875.5, 1271.4)	(1074.0, 1063.1)
RN	(1053.7, 1087.2)	(1074.5, 1064.3)	(1111.4, 1049.0)	(1060.6, 1077.3)	(1071.2, 1074.5)	(910.3, 1233)	(1034.8, 1087.2)
AH	(1041.0, 1117.9)	(1052.7, 1112.2)	(1110.7, 1099.2)	(1054.3, 1132.2)	(1049.0, 1133.7)	(905.0, 1310.2)	(1204.5, 948.6)
NN + 2opt	(1086.8, 1062.2)	(1064.8, 1076.9)	(1144.4, 1043)	(1066.7, 1059.1)	(1088.3, 1074.9)	(893.2, 1258.4)	(1024.3, 1132.7)
RN + 2opt	(1064.6, 1075.0)	(1087.7, 1046.4)	(1133.5, 1042.8)	(1064.3, 1103.6)	(1089.0, 1076.3)	(921.1, 1231.3)	(1065.4, 1087.9)
Hyper	(1278.2, 862.5)	(1233.7, 913.5)	(1350.6, 858.2)	(1281.0, 855.7)	(1230.1, 914.2)	(1092.5, 1093.6)	(1224.0, 972.8)
Insert + 2opt	(1066.3, 1075.6)	(1077.4, 1058.3)	(979.8, 1165.2)	(1089.6, 1044.3)	(1094.5, 1059.9)	(963.3, 1237.2)	(1091.5, 1089.1)

Table 4 Simulation results of Agent 1 adopting different heuristics in 1000 CTSPs

Heuristics	*Payoffs*	*Visited cities*	*Wasted visits*	*Distance travelled*
NN	4317.2	102.6	3.29	650.2
RN	4293.2	102.3	2.93	648.8
AH	4216.7	97.0	0	570.7
NN + 2-opt	4395.1	104.6	3.43	674.9
RN + 2-opt	4384.6	105.2	3.76	677.6
Hyper	5374.6	123.3	0.01	803.4

Table 5 The rate that Hyper adopts different low level heuristics

Low-level heuristics	*NN*	*RN*	*AH*	*NN* + 2-*opt*	*RN* + 2-*opt*
The frequency that Hyper applies this heuristic (%)	41.3	26.7	42.6	43.1	36.5

consider the strategy of other agents, so occasionally leads to a worse subtour. This has limited influence on the hyper-heuristic because it chooses low-level heuristics on each move and the negative effect of a bad move is short lived. Simulation results show that Hyper performs well in interacting with this heuristic.

4.2 CTSP with three agents and 300 cities

A total of 300 cities located in a square area of 100X100 are randomly placed. The payoff of visiting each city is set to be 50 while the cost of travel is equivalent to the distance travelled. Three agents are identical and they are initially located in three different cities.

We ran 1000 CTSPs and computed the average payoffs of adopting different heuristics. The simulation results are shown in Tables 4 and 5 and Figure 6 (we only show the payoffs of adopting Hyper and NN in Figure 6 as the payoffs of other heuristics are similar to NN). In Table 4, the values are average values for Agent 1 adopting different heuristics over 1000 CTSPs. Table 5 shows how frequently each low-level heuristic is adopted by Hyper. Note that the aggregated value of the rates is >1 because some heuristics frequently create the same moves. If two or more heuristics create the same move that leads to the highest payoff, all these heuristics are considered to be adopted in that move. Figure 6 shows the payoffs of Agent 1 given the combination of other agents' heuristics. The horizontal axes denote the heuristics chosen by Agents 2 and 3, respectively. It shows that Hyper receives higher payoff than other heuristics.

The reason for Hyper's good performance is that it inherits the good features of the low-level heuristics. For example, AH can avoid wasted visits in most situations while the other heuristics cannot. Hyper achieves a low rate of wasted visits by adopting AH in the situations where the agent is close to other agents.

NN and AH create the same moves in most situations. We have implemented a hyper-heuristic without NN and it performs similar to Hyper. This suggests that it is not required to establish a large set of low-level heuristics and we only need to consider the heuristics that lead to good quality solutions.

5. Conclusions and future work

CTSP is different from other variants of TSP because it is not only a NP-hard search problem but also a decision-making problem. CTSP is a combination of optimization and decision making. It is difficult to compute the optimal strategies of the agents in a CTSP by means of an exhaustive search because there is uncertainty involved, although the structure of the problem (the payoffs, the states of the agents and cities) is exactly known. CTSP could be an abstract model of some complicated decision-making problems in the real world.

A hyper-heuristic model for CTSP has been proposed where the problem is greatly simplified and simple heuristics can be applied to compute approximate solutions. The two-level structure of the hyper-heuristic makes it possible to combine approaches from different fields. The proposed

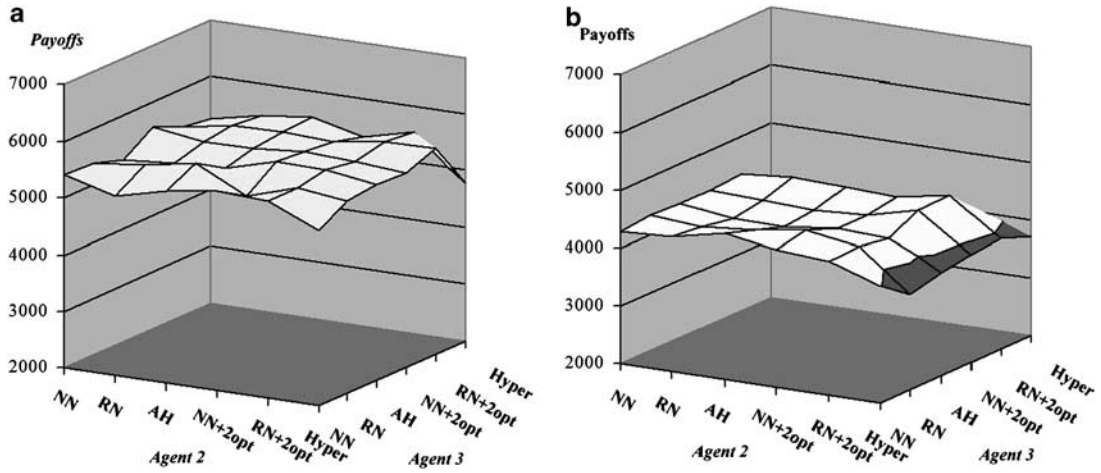

Figure 6 Payoffs of Agent 1 when the agent adopts (a) Hyper-heuristic (b) NN.

hyper-heuristic consists of a high-level heuristic that is based on game theory and a number of low-level heuristics that are based on TSP heuristics. We have conducted a preliminary hyper-heuristic study for the CTSP, which demonstrates the potential of applying hyper-heuristics to complex games.

The simulation results show that the proposed hyper-heuristic performs well in computing approximate solutions for CTSP. The hyper-heuristic is able to inherit the good features of the low-level heuristics. It demonstrates the potential of developing much efficient heuristics for CTSP by using existing heuristics.

There are many directions for further work. First, it would be non-trivial to develop some quality, stability, and robustness measures for the solutions to CTSP in the absence of Nash equilibrium. Second, the low-level heuristics we adopted are not 'good' heuristics for CTSP. There needs some better constructive heuristics to achieve an even more efficient hyper-heuristic. Third, the amount of computation should be considered when a large number of low-level heuristics are adopted.

Acknowledgements—This study is supported by an EPSRC (Engineering and Physical Science Research Council) project (Ref: EP/D061571/1). The authors would like to thank Dr Andrew Parkes for his helpful comments. The authors also thank two anonymous reviewers for their helpful comments.

References

Applegate D, Bixby R, Chvátal V and Cook W (2006). *The Travelling Salesman Problem: A Computational Study*. Princeton University Press: Princeton, NJ.

Bai R, Burke E and Kendall G (2008). Heuristic, meta-heuristic and hyper-heuristic approaches for fresh produce inventory control and shelf space allocation. *Journal of the Operational Research Society* **59**(10): 1387–1397.

Bektas T (2006). The multiple traveling salesman problem: An overview of formulations and solution procedures. *Omega* **34**(3): 209–219.

Burke E, Kendall G and Soubeiga E (2003a). A tabu-search hyperheuristic for timetabling and rostering. *Journal of Heuristics* **9**(6): 451–470.

Burke E, Kendall G, Newall J, Hart E, Ross P and Schulenburg S (2003b). Hyper-heurisitics: An emerging direction in modern search technology. In: Glover G and Kochenberger K (eds). *Handbook of Metaheuristics*. Chapter 16, Kluwer Academic Publishers: The Netherlands, pp 457–474.

Burke E, McCollum B, Meisels A, Petrovic S and Qu R (2005). A graph-based hyper-heuristic for educational timetabling problems. *European Journal of Operational Research* **176**(1): 177–192.

Burke EK, Hyde MR and Kendall G (2006). Evolving bin packing heuristics with genetic programming. In: *Proceedings of the 9th International Conference on Parallel Problem Solving from Nature (PPSN 2006)*. Lecture Notes in Computer Science, Vol. **4193**, pp 860–869.

Burke E, Hyde M, Kendall G, Ochoa G, Ozcan E and Woodward J (2009). Exploring hyperheuristic methodologies with genetic programming. In: Mumford C and Jain L (eds). *Collaborative Computational Intelligence*. Springer Verlag: Berlin, Heidelberg, pp 177–201.

Burke E, Hyde M, Kendall G and Woodward J (2010a). A genetic programming hyper-heuristic approach for evolving 2-D strip packing heuristics. *IEEE Transactions on Evolutionary Computation* **14**(6): 942–958.

Burke E, Hyde M, Kendall G, Ochoa G, Ozcan E and Woodward J (2010b). A classification of hyper-heuristic approaches. In: *Handbook of Meta-Heuristics*. Kluwer: The Netherlands, pp 449–468.

Cowling P, Kendall G and Soubeiga E (2000). A hyper-heuristic approach to scheduling a sales summit. Selected papers from the *Third International Conference on Practice and Theory of Automated Timetabling, PATAT 2000*; Konstanz, Germany, pp 176–190.

Garrido P and Riff M (2007). DVRP: A hard dynamic combinatorial optimisation problem tackled by an evolutionary hyper-heuristic. *Journal of Heuristics* **16**(6): 795–834.

Gendreau M and Potvin J (2005). Metaheuristics in combinatorial optimization. *Annals of Operations Research* **140**(1): 189–213.

Gutin G and Punnen A (2002). *The Travelling Salesman Problem and Its Variations*. Kluwer Academic Publishers: The Netherlands.

Laporte G (2010). A concise guide to the traveling salesman problem. *Journal of Operational Research Society* **61**(1): 35–40.

Lawler E, Lenstra J, Kan A and Shmoys D (1986). *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. John Wiley & Sons: Hoboken, NJ.

Nash J (1951). Noncooperative games. *Annals of Mathematics* **54**(2): 286–295.

Osborne M and Rubinstein A (1999). *A course in Game Theory*. MIT Press: New York.

Papadimitriou C and Roughgarden T (2005). Computing equilibria in multi-player games. In: *Proceedings of 16th ACM-SIAM Symposium on Discrete Algorithms*, Philadelphia, PA, pp 82–91.

Pisinger D and Ropke S (2007). A general heuristic for vehicle routing problems. *Computers and Operations Research* **34**(8): 2403–2435.

Ross P (2005). Hyper-heuristics. In: Burke EK and Kendall G (eds). *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*. Springer Science + Business Media, LLC, pp 529–556.

Received March 2011;
accepted February 2012 after two revisions