Journal of the Operational Research Society (2005) 56, 816–824

www.palgrave-journals.com/jors

Tabu search embedded simulated annealing for the shortest route cut and fill problem

A Lim[1], B Rodrigues[2]* and J Zhang[3]

[1]*The Hong Kong University of Science and Technology, Kowloon, Hong Kong;* [2]*Singapore Management University, Singapore; and* [3]*National University of Singapore, Singapore*

The shortest route cut and fill problem proposed by Henderson *et al*[1] is studied in this paper where we extend the model to include multiple vehicles and a makespan objective. A new tabu search embedded simulated annealing algorithm for both models is developed. Computational experiments show that the new approach is robust and achieves better solutions when compared with those found using Henderson *et al*'s algorithm for larger test cases within significantly shorter times.
Journal of the Operational Research Society (2005) **56,** 816–824. doi:10.1057/palgrave.jors.2601900
Published online 1 December 2004

Keywords: cut and fill; tabu search; simulated annealing

Introduction

It is common to excavate earth from certain locations and move it to other locations within construction sites. Heavy earth-moving vehicles are required for this purpose, which incur high cost to operate and maintain. Finding routes that minimize the distances travelled by vehicles is therefore an important planning goal. The problem for a single vehicle is called shortest route cut and fill problem (SRCFP) and has recently been studied by Henderson *et al.*[1]

The SRCFP is related to the well-known symmetric travelling salesman problem[2] and the vehicle routing problem,[3] and is a special case of the capacitated travelling salesman with pickups and deliveries problem (CTSPDP) when the vehicle capacity is set to one and all goods are identical since the latter consists of two location types.[1] The problem is also a variation of the capacitated vehicle routing problem.[4] The CTSPDP has been studied by Anily and Bramel[5] who developed polynomial approximation algorithms, and by Renaud *et al*[6] who adapted classical heuristic methods and proposed a new efficient heuristic for the problem.

The SRCFP has been shown to be **NP**-hard,[1] so that a complete search for the problem within reasonable computational times is unlikely. This motivated the development of local search or heuristic methods. In Henderson *et al*,[1] greedy heuristic and a simulated annealing algorithm was used for the problem. In this paper, we develop a specialized tabu search (TS) embedded simulated annealing algorithm to overcome the shortcomings of a purely simulated annealing approach. Also, we examine the case when more than one vehicle is allowed to reflect more realistic situations and consider minimizing the makespan as an objective in the problem.

This work is organized as follows. The next section provides a description and formulations of the problem. In the third section, the algorithm proposed by Henderson *et al*[1] and the algorithm we developed are presented. In the fourth section, we extend the solution approach to address situations where multiple vehicles are used. Computational experiments are described and results analysed in the next section and the final section summarizes the work.

Problem description and formulation

The SRCFP is defined on a site comprising two types of locations called *cut* and *fill* locations. A cut location has an amount of excess earth to be excavated while a fill location needs to be filled in with earth. In order to describe locations, a site is represented by a rectangular grid, where any irregular-shaped site can be replaced by the smallest rectangle enclosing it. Each rectangle in the grid corresponds to one location and must be one of the three possible location types: cut, fill or balanced. Here, a balanced location is one with no excess or shortfall of earth.

When there is only one earth-moving vehicle available, the vehicle can load up a fixed amount of earth at a cut location, travel to a fill location and deposit its load there. The distance between two locations is taken as the Euclidean distance between the locations and a unit of earth is taken to be the capacity of the vehicle. We define the amount of excess earth at a cut location to be in unit cuts and the amount of earth in shortfall at a fill location to be in unit fills. Usually, the total number of unit cuts is not equal to the

*Correspondence: B Rodrigues, School of Business, Singapore Management University, 469 Bukit Timah Road, Singapore 259756, Singapore.
E-mail: br@smu.edu.sg

total number of unit fills so that a dummy location is created. Whether the dummy location is to be a cut or fill location and what number of units of earth is assigned to it are determined by taking total surplus to equal total deficit. Once this is determined, the dummy location becomes a part of the problem with the other locations.

Starting at a cut location, the vehicle visits cut and fill locations alternately and finally returns to its starting point. Which location is taken as the starting point is immaterial since the route forms a circuit. At each cut location, it loads a unit of earth, and travels to a fill location where it deposits the unit of earth. The vehicle then heads to a cut location with excess earth yet to be moved. This process is repeated until all excess earth has been moved to fill locations. The vehicle can visit the same location more than once if there is more than one unit cut (or fill) at that location. To simplify the problem, all visits can be taken to be distinct and a visit to a cut location will remove a single unit cut while a visit to a fill location deposits a single unit fill. Some locations will receive no visit if they are initially balanced. As a result of the dummy location, the number of visits to cut locations must equal visits to fill locations. Finally, the vehicle must return to the starting point. The objective is then to find a route with the shortest total distance travelled by the vehicle.

We note here that the problem is similar to the travelling salesman problem except that the vehicle must alternate between cut and fill locations and can visit the same location more than once.

We use notations and definitions provided in Henderson *et al*[1] for continuity. Let $G=\{g_1,g_2,\ldots,g_n\}$ be a set of n locations (excluding balanced locations), each associated with a coordinate and the number of unit cuts or unit fills. We use a positive integer to represent unit cut volume and a negative integer for unit fill. Define $L=\{l_1,l_2,\ldots,l_k\}$ $(k\geqslant n)$ to be the set of single unit cut and fill locations corresponding to locations in G. As mentioned, there may be more than one unit cut or fill at any location. Suppose location g_i is a cut location with x unit cuts. Then, there will be x locations in L, say $l_j,l_{j+1},\ldots,l_{j+x-1}$, corresponding to g_i. Each of these has volume $+1$. L can be further divided into two subsets: $L^+=\{l_1^+,l_2^+,\ldots,l_m^+\}$ contains all single unit cut locations and $L^-=\{l_1^-,l_2^-,\ldots,l_m^-\}$ contains all single unit fill locations, where m is the total number of units of earth to be moved. The relationship between these is: $|L|=2|L^+|=2|L^-|=2m=k$. Define $c_i\in L^+$ to be a single cut location for $i=1,2,\ldots,m$, and $f_i\in L^-$ to be a single fill location for $i=1,2,\ldots,m$, and let $D(l_i^+,l_j^-)$ be the distance between location l_i^+ and l_j^- for $i,j=1,2,\ldots,m$ (which is symmetric, ie, $D(l_i^+,l_j^-)=D(l_j^-,l_i^+)$). Finally, let $R=(c_1,f_1,c_2,f_2,\ldots,c_m,f_m)$ represent a route traversed by the vehicle, taken to be a Hamiltonian circuit with the condition that points on the route alternate between cut and fill locations.

The SRCFP can now be defined as follows: given a set of unit cut locations $L^+=(l_1^+,l_2^+,\ldots,l_m^+)$, and a set of unit fill locations $L^-=\{l_1^-,l_2^-,\ldots,l_m^-\}$ and a symmetric matrix D representing distances between cut and fill locations, find a route $R=(c_1,f_1,c_2,f_2,\ldots,c_m,f_m)$, $c_i\in L^+$, $f_i\in L^-$, $i=1,2,\ldots,m$, which minimizes $\sum_{i=1}^{m=1}[D(c_i,f_i)+D(f_i,c_{i+1})]+D(c_m,f_m)+D(f_m,c_1)$.

It was shown in Henderson *et al*[1] that the cardinality of the solution space to this problem grows exponentially as the number of unit cut and fill locations increase and that the problem is **NP**-hard

Formulation as a travelling salesman problem

In order to benchmark solutions obtained by their algorithms, Henderson *et al*[1] used an integer programming formulation of the SRCFP based on a model by Miller, Tucker and Zemlin[7] for the travelling salesman problem:

$$\text{minimize} \sum_{(i,j)\in A} D_{ij}x_{ij} \tag{1}$$

$$\sum_{j\in L^-} x_{ij}=1, \quad i\in L^+ \tag{2}$$

$$\sum_{j\in L^+} x_{ij}=1, \quad i\in L^- \tag{3}$$

$$\sum_{i\in L^-} x_{ij}=1, \quad j\in L^+ \tag{4}$$

$$\sum_{i\in L^+} x_{ij}=1, \quad j\in L^- \tag{5}$$

$$p_i+1-k(1-x_{ij})\leqslant p_j, \quad (i,j)\in A, j\neq 1 \tag{6}$$

$$p_1=1 \tag{7}$$

$$x_{ij}+x_{ji}\leqslant 1, \quad (i,j)\in A, i<j \tag{8}$$

$$x_{ij}\in\{0,1\}, \quad (i,j)\in A \tag{9}$$

$$p_i\in\{1,2,\ldots,k\}, \quad i=1,2,\ldots,k \tag{10}$$

In the above model, L, L^+, L^- and k are as defined and A is the set of arcs from unit cut locations to unit fill locations and vice versa. D_{ij} is the Euclidean distance between location i and j, which cannot be both cut or fill locations. The decision variable x_{ij} is 1 if location i immediately precedes j, and 0 otherwise, and the decision variable p_i is the position of location l_i in the Hamiltonian circuit, where the starting location is defined to be at position 1.

In their work, Henderson *et al*[1] used ILOG OPL Studio 3.0 to generate the integer programs for test problems and CPLEX 6.6 to solve these problems. They reported that CPLEX was able to solve randomly generated 5×5 problems but was terminated after 4 h for 7×7 cases. Since CPLEX could not solve larger problems, the Held-Karp[8,9] 1-tree lower bound was computed for each of these problems. A description of this lower bound is given in Henderson *et al.*[1]

The algorithms

Henderson et al's Algorithm

Simulated Annealing (SA)[10] was used by Henderson *et al.* The outline of their SA algorithm is provided in Algorithm 1 for easy reference.

Algorithm 1 Henderson *et al.*'s Algorithm

Pick an initial solution ω
Set temperature change counter $t=0$ and number of temperature changes τ
Select a temperature cooling schedule, $T(t)$, and an initial temperature $T=T(0)\geqslant 0$
Set a repetition schedule $N(t)$, and number of iterations at each temperature $T(t)$
Let η be a function generates a neighbour of a solution.
Let f be a function evaluates the objective value of a solution.
repeat
  Set repetition counter $q=0$
  repeat
    Generate a solution $\omega'=\eta(\omega)$ and calculate $\delta=f(\omega')-f(\omega)$
    if $\delta\leqslant 0$ **then**
      $\omega\leftarrow\omega'$
    else
      $\omega\leftarrow\omega'$ with probability $e^{-\delta/T(t)}$
    end if
    $q\leftarrow q+1$
  until $q=N(t)$
  $t\leftarrow t+1$, $T\leftarrow T(t)$
until Stopping criterion is met when $q=N(\tau)$

An initial solution is found by randomly picking unit cut and fill locations alternately. For each execution of the SA search, the temperature changes $\tau=500$ times and at each temperature, there are $N(t)=4m$ iterations for $t=1,2,\ldots,\tau$. The initial temperature is set to $0.4mM$, where $M=\max\{D(l_i,l_j),\ i,j=1,2,\ldots,m\}$ and for each temperature change, the original temperature is multiplied by the cooling factor $\beta=0.98$.

Given the current solution, a neighbourhood function η will randomly produce a new solution. Suppose the original route is $R=(c_1,f_1,c_2,f_2,\ldots,c_m,f_m)$. To generate a neighbourhood of R, $c_i,c_j\in\{c_1,c_2,\ldots,c_m\}$, $c_i\neq c_j$ are randomly selected and the subsequence of points between c_i and c_j (inclusive) is reversed, that is, for $i,j=1,2,\ldots,m$, $i<j$,

$$\eta(R) = \{c_1,f_1,\ldots,f_{i-1},c_j,f_{j-1},c_{j-1},\ldots f_i,c_i,f_j\ldots,c_m,f_m\} \quad (11)$$

Since the distance matrix is symmetric, only two distances will be affected by this reversal, that is the two distances related to c_i and c_j.

In the implementation, 50 replications were executed for every problem instance, and the mean, standard deviation, minimum and maximum of the objective function values over the replications are reported.

A TS embedded simulated annealing algorithm

A weakness of SA can be that it is completely memoryless. The approach we take is motivated by probabilistic tabu search (TS),[11] which exploits the asymptotic optimality of SA and the memory features of TS.[12] Owing to the random nature of a purely SA approach, potentially good solutions can often be missed, and convergence slow. In order to address this weakness, TS can used for better in-depth search and to accelerate the search process. Here, we construct a tabu-embedded simulated annealing algorithm where once the SA component generates a random neighbourhood for the current solution, TS is applied to improve the solution. On the other hand, SA generates random neighbours, which help the TS component avoid becoming trapped at local optima. An outline of this combined approach is given in Algorithm 2 which we call TSSA below.

The initial temperature, $T(0)$, used is same as that used by Henderson *et al*,[1] but the cooling factor was set at $\beta=0.95$. We note that the cooling factor, β, used by Henderson *et al* was 0.98. In the experiments, we found that $\beta=0.98$ gave the best results for their algorithm. In our algorithm, results obtained using $\beta=0.95$ were as good as those using $\beta=0.98$, but we chose $\beta=0.95$ since it allowed for faster running times. The number of temperature changes τ and number of iterations at each temperature N are as those used by Henderson *et al*[1] and the initial solution used was similarly constructed.

Algorithm 2 The TSSA Algorithm

Initialize temperature $T=T(0)$ and cooling ratio β
Generate an initial solution ω
Set total number of temperature changes τ, and counter $t=0$
Set the number of iterations at each temperature to N
Let η' be a function generates a neighbour of a solution.
repeat

Set repetition counter $q=0$
repeat
  Generate a neighbour $\omega'=\eta'(\omega)$ and $\delta=f(\omega')-f(\omega)$
  if $\delta\leqslant 0$ **then**
    $\omega\leftarrow\omega'$
  else
    $\omega\leftarrow\omega'$ with probability $e^{-\delta/T(t)}$
    $q\leftarrow q+1$
  end if
until $q=N$
$t\leftarrow t+1$, $T\leftarrow\beta T$
$\omega\leftarrow TS(\omega)$
until $t=\tau$

A key in implementing SA is the neighbourhood function that is used.[13] Since neighbourhoods impose a topology on the problem, one where local optima are 'shallow' rather than one that gives many 'deep' local optima is often preferred.[14] Consequently, we chose to use a new neighbourhood function η' in the SA and a TS function TS as indicated in the algorithm.

Given a route $R=(c_1,f_1,c_2,f_2,\ldots,c_m,f_m)$, $i,j\in\{1,2,\ldots,m\}$ with $i<j$ are selected randomly. A third $k\notin\{i,j\}$ is then randomly selected and a neighbour is randomly generated by either shifting sequence $(c_i,f_i,\ldots,c_j,f_j)$ immediately before c_k, or shifting sequence $(f_i,c_{i+1},\ldots,f_{j-1},c_j)$ to be immediately after c_k. The neighbourhood function resulting from this is then given by the following:

$$\begin{aligned}\eta'(R)=\{&c_1,f_1,\ldots,c_{k-1},f_{k-1},c_i,f_i,\ldots,c_j,f_j,c_k,\\&f_k,\ldots,c_m,f_m\}\quad\text{or}\quad\{c_1,f_1,\ldots,c_k,f_i,\\&c_{i+1},\ldots,f_{j-1},c_j,f_k,\ldots,c_m,f_m\}\\&\text{for}\quad i,j=1,2,\ldots,m,i<j,k\notin\{i,j\}\end{aligned}\tag{12}$$

We note that η' acts on four points in the route (the original two ends of the shifted segment as well as the new two ends), whereas η acts only on two.

In TS, the neighbourhood of the current solution is searched exhaustively, and the best solution picked to replace the current solution. A tabu list is maintained to prevent undoing recent operations. In the algorithm here, a neighbouring solution is generated by swapping two unit cut (or fill) locations as follows: given a route $\{c_1,f_1,\ldots,c_i,\ldots,c_j,\ldots,c_m,f_m\}$, pick c_i, c_j, $0\leqslant i<j\leqslant m$ randomly and take the neighbour to be $\{c_1,f_1,\ldots,c_j,\ldots,c_i,\ldots,c_m,f_m\}$.

Attempting all possible swaps may not lead to good solutions since there will be many duplicate swaps. As mentioned previously, a cut (fill) location can appear more than once in the route if it contains many unit cuts (fills). Swaps relevant to duplicate locations will result in wasted time and, therefore, instead of searching the entire neighbourhood, a number of random swaps were made (200, say).

Extensions

We examine here extensions to the SRCFP, which include the use of multiple vehicles and minimizing the total time travelled by vehicles in the objective.

Multiple vehicles

As a first step towards a more realistic problem, it is natural to allow for multiple vehicles in the model. Indeed, similar extensions have been addressed in related problems such as the vehicle routing problem, where more than one vehicle is allowed. Construction sites often employ different vehicle types suited for varying routes/terrain, while vehicle characteristics make them unusable on certain routes. Also, certain vehicles can be cheaper to operate on specific routes.

In the multiple-vehicle model, the objective in the SRCFP remains as before. The solution algorithm, however, is adapted, where we allow multiple homogeneous vehicles and where the objective is to find the shortest route for every vehicle used to minimize the total distance travelled by all vehicles.

The algorithm for the multiple-vehicle model is similar to that for the single-vehicle one except in the following aspects. First, the simulated annealing component picks a random neighbour solution by shifting a sequence of locations from one route to another instead of shifting within the same route. Second, the TS component is allowed to swap locations either for the same vehicle or between the different vehicles available. The simulated annealing neighbourhood is restricted to only inter-route movements because this produces significant changes to the solution, while fine-tuning is left to TS.

It can be expected that cost is improved by using more vehicles, as for example, in a construction site with cut and fill locations concentrated in the upper-left and lower-right corners and with the remaining locations balanced. With one vehicle, a good solution is to complete all work in one corner, and then travel (a longer distance) to the other corner to complete work there. However, with two vehicles, each corner can be serviced by a single vehicle and the longer trip avoided.

Minimizing makespan

Yet another extension of the multiple-vehicle model is to minimize the *makespan* ($C_{\max}$) instead of the total distance travelled. For any problem, the solution consists of one route R_i for each vehicle i with the objective to minimize $C_{\max}=\max R_i$. We can expect here that a good solution will have routes with approximately the same length, and the

total distance travelled to be close to that achieved for the original problem.

Computational experiments and analysis

In all experiments, we used a Pentium 3800 MHz machine with 1 Gb of memory and code written in Java with Linux OS.

Data generation

The test cases were randomly generated as follows: given the side length of the construction site, n, a $(n+1)\times(n+1)$ matrix was constructed with locations at (i,j), $i,j=1,2,\ldots,n$ in a rectangular grid and where $(0,1)$ is designated the dummy location. The amount of earth for each location is uniformly distributed in $[-3,3]$, and 10 test cases were generated for each $n=5,7,9,15$ where the first three sizes $n=5,7,9$ were used by Henderson *et al.*[1] However, these are relatively small and inadequate for testing the performance of the algorithm for more realistic situations. Hence, an additional set of test cases, for $n=15$, was included.

We note here that the volume of earth generated at each location determines the size of the problem instance, that is, the number of unit cut and fill locations. Given a 9×9 problem, for example, the maximum possible number of earth units (both excess and shortfalls) is 243, and the mean approximately 139. In Henderson *et al*'s[1] paper, the mean was taken to be a half of the maximum 121.5, where, in fact, the mean should be larger than this. For any location, there can be 0, 1, 2 or 3 units of earth where the probability of a location being balanced is lower than otherwise. The former probability is 1/7, while the latter adds to 2/7 so the mean units of earth for any location is 12/7 and the mean number of units of earth for the entire site is $12n^2/7$. This value was used to decide on the appropriate number of total unit cut and fill locations for the problem.

The single vehicle case

Both Henderson *et al*'s[1] and the TSSA algorithms were implemented. Henderson *et al* calculated the Held-Karp 1-tree lower bound and used Lagrange multipliers to improve these. According to their paper, solutions from their algorithm were within 0.7% from the lower bound, on average.

Table 1 shows the results obtained by both algorithms for problem size $n=5,7,9$. All times in the tables are in seconds.

In all, 10 replications were executed for each of the problem instances. The mean (μ), minimum (min), maximum (max) and standard deviation (σ) values are included in the table, together with the execution time (total time to complete all replications) and lower bound (LB) denotes the Held-Karp 1-tree lower bound. As we can see, results for mean values are almost identical for both algorithms. However, the execution time for TSSA tended to be shorter as the problem size increased. It also had a lower average value over these test cases with 104.47 s compared with the average of 114.37 for Henderson *et al*'s algorithm. Also, the TSSA had smaller standard deviations with an average value of 0.57 compared with Henderson *et al*'s 0.78.

To further verify the speed and robustness of TSSA, we tested the algorithms on a 15×15 test set where results are given in Table 2.

For this large test set, the TSSA was significantly faster than the Henderson *et al* algorithm with an average running time of 987.80 s compared with 2166.40 s. The TSSA algorithm obtained consistently smaller standard deviations with an average value of 3.44 compared with 5.02 for Henderson *et al*'s algorithm. When we compared the average of the differences between minimum values (min) obtained by the algorithms and the lower bounds (LB), we found that the TSSA obtained an average value of 34.47 compared to 43.83 obtained by the Henderson *et al* algorithm, which is an approximately 21% improvement.

The multiple vehicle case

The multiple vehicle algorithm was applied to the same sets of problem instances with the number of vehicles equal to 2 and 4 and the best results (in column $\sum R_i$) together with execution times are given in Table 3.

From the table, we find that using more vehicles can improve the objective function value as can be expected. However, the improvement is not significant. Since problem instances were randomly generated with uniformly distributed random variables, all cut and fill locations are uniformly spread across the site. The example described in the preceding section is therefore unlikely. In this case, we can conclude that using more vehicles does not help much because the work to be performed remains almost the same regardless of how many vehicles there are.

In the real world, however, there would be situations without uniform spreads of locations where it can happen that using multiple vehicles can reduce costs. A good approach for these situations would be to use various number of vehicles to find the most cost-effective number to use. The situation where different types of vehicles are required is catered for in the multiple-vehicle model. We note that execution times tend to be shorter as the number of vehicles increased.

After extending the problem to minimize makespan, we applied the algorithm to the same set of test cases with number of vehicles equal to 2 and 4. Table 4 provides results that are compare with original values where column, $C_{\max}$, gives the minimum makespan obtained. The total distance travelled, $\sum R_i$, for both makespan and original values are also provided.

Table 1 Comparisons for the 5×5, 7×7 and 9×9 cases (single vehicle)

		Henderson et al Algorithm					*TSSA*				
Test set	*LB*	*min*	μ	*max*	σ	*time*	*min*	μ	*max*	σ	*time*
5-1	48.13	48.89	49.30	49.95	0.38	19	48.89	48.94	48.96	0.03	27
5-2	76.70	77.52	77.93	78.70	0.32	26	77.52	77.52	77.52	0.00	37
5-3	53.07	54.31	55.28	57.20	0.77	23	54.31	54.93	55.47	0.39	36
5-4	179.03	180.16	180.18	180.69	0.08	25	180.16	180.16	180.16	0.00	26
5-5	110.10	110.34	110.64	111.34	0.32	27	110.34	110.36	110.37	0.01	31
5-6	64.80	64.97	65.42	66.31	0.36	26	64.97	65.23	65.48	0.18	34
5-7	57.10	58.31	58.50	59.91	0.28	18	58.31	58.31	58.31	0.00	28
5-8	124.52	124.87	124.94	125.64	0.21	27	124.87	124.87	124.87	0.00	36
5-9	89.00	89.25	89.74	91.34	0.51	24	89.25	89.66	89.99	0.25	30
5-10	144.35	144.87	144.91	145.29	0.07	29	144.87	144.87	144.87	0.00	35
7-1	267.40	268.20	269.52	271.13	0.73	115	268.38	269.36	270.34	0.64	116
7-2	102.88	103.12	104.59	106.11	0.63	61	103.81	104.34	104.87	0.38	66
7-3	137.68	137.86	138.65	139.99	0.51	67	137.93	138.31	138.55	0.22	75
7-4	257.53	258.71	259.73	262.54	0.86	97	258.59	258.80	259.16	0.19	103
7-5	105.18	106.35	107.68	109.53	0.70	69	106.28	107.17	108.18	0.59	72
7-6	126.93	127.95	129.55	133.79	1.20	73	127.95	128.51	129.05	0.33	78
7-7	148.55	149.86	151.29	153.40	0.84	84	149.80	150.56	151.62	0.73	88
7-8	271.35	271.97	272.74	274.47	0.52	130	272.05	272.62	273.74	0.54	132
7-9	127.35	128.91	130.04	132.29	0.80	87	128.65	129.38	130.21	0.64	84
7-10	109.07	109.73	111.43	113.86	0.94	88	109.73	111.11	112.36	0.84	86
9-1	198.10	200.52	204.73	211.48	2.21	222	201.41	204.13	207.67	1.89	186
9-2	593.11	596.87	599.40	604.26	1.38	280	596.72	598.52	600.17	1.13	228
9-3	283.56	286.71	289.43	291.89	1.24	248	286.17	287.84	289.58	1.14	198
9-4	233.18	234.55	237.09	240.11	1.23	172	233.79	236.07	237.91	1.40	151
9-5	237.13	238.55	242.13	244.33	1.19	237	239.39	241.18	243.45	1.22	191
9-6	247.86	249.93	252.51	254.98	1.18	251	250.45	251.61	253.06	0.88	201
9-7	444.79	447.40	448.41	450.42	0.69	239	447.36	447.75	448.56	0.32	215
9-8	220.61	223.11	225.48	228.78	1.19	224	222.61	225.24	227.09	1.57	177
9-9	298.68	300.68	302.23	305.74	0.98	239	300.34	301.41	302.59	0.79	196
9-10	368.05	369.91	371.58	373.31	0.99	204	369.06	370.63	372.07	0.88	171

Table 2 Comparisons for the 15×15 case (single vehicle)

		Henderson et al Algorithm					*TSSA*				
Test set	*LB*	*min*	μ	*max*	σ	*time*	*min*	μ	*max*	σ	*time*
15-1	734.47	778.91	787.22	801.20	4.77	2117	768.16	772.97	776.80	2.91	971
15-2	603.94	650.56	662.86	674.67	5.37	1955	646.43	652.96	657.97	4.12	893
15-3	1151.91	1178.21	1189.61	1198.83	3.95	2068	1168.48	1173.08	1176.81	2.32	958
15-4	894.23	939.91	948.56	957.49	4.17	2170	930.60	935.44	939.55	2.85	1012
15-5	579.56	628.75	639.62	651.38	5.60	2060	619.38	629.06	639.16	5.99	932
15-6	1053.75	1082.65	1090.25	1097.77	3.42	1968	1070.78	1074.70	1079.99	2.40	926
15-7	794.51	842.47	852.90	871.09	5.89	2252	830.30	837.32	842.25	3.73	1034
15-8	987.57	1030.67	1044.12	1055.90	5.36	2393	1023.44	1027.74	1031.68	2.63	1067
15-9	638.32	695.21	709.34	722.45	6.26	2419	687.50	694.98	699.57	3.96	1068
15-10	688.35	737.59	748.14	757.96	5.39	2262	726.22	733.37	738.02	3.50	1017

As we see, the longest route makespan is very close to $(\sum R_i)/v$, which tells us that all routes have approximately the same length. Furthermore, for most cases, the total distance travelled falls within 5% from the best we have found.

Summary

In this paper, the shortest route cut and fill problem proposed by Henderson *et al*[1] was studied. A new TS search embedded simulated annealing algorithm was developed to

Table 3 Single *versus* multiple vehicles

	$v=1$		$v=2$		$v=4$	
Problem	$\sum R_i$	*time*	$\sum R_i$	*time*	$\sum R_i$	*time*
5-1	48.89	27	48.13	27	48.13	27
5-2	77.52	37	76.70	37	76.29	37
5-3	54.31	36	53.79	32	52.55	32
5-4	180.16	26	179.06	28	178.26	27
5-5	110.34	31	110.16	31	109.89	30
5-6	64.97	34	64.97	34	64.97	34
5-7	58.31	28	57.51	28	56.51	27
5-8	124.87	36	124.52	37	124.48	36
5-9	89.25	30	89.07	30	89.00	29
5-10	144.87	35	144.36	35	144.25	33
7-1	268.38	116	267.82	113	267.82	115
7-2	103.81	66	103.45	65	103.05	67
7-3	137.93	75	137.31	73	136.55	75
7-4	258.59	103	257.94	101	257.78	100
7-5	106.28	72	106.35	71	104.94	70
7-6	127.95	78	127.10	77	126.90	77
7-7	149.80	88	150.09	86	148.31	85
7-8	272.05	132	271.29	128	270.71	126
7-9	128.65	84	128.61	82	126.97	79
7-10	109.73	86	109.73	86	109.73	85
9-1	201.41	186	202.18	178	201.83	178
9-2	596.72	228	595.59	221	595.99	214
9-3	286.17	198	286.22	195	286.22	192
9-4	233.79	151	234.33	150	234.50	148
9-5	239.39	191	240.04	190	239.26	189
9-6	250.45	201	248.97	198	249.01	193
9-7	447.36	215	446.67	210	445.71	205
9-8	222.61	177	221.80	175	222.67	174
9-9	300.34	196	300.32	190	300.08	185
9-10	369.06	171	369.05	163	369.25	163
15-1	768.16	971	764.07	936	759.66	907
15-2	646.43	893	646.72	870	640.28	840
15-3	1168.48	958	1169.28	923	1164.13	882
15-4	930.60	1012	928.50	974	924.74	939
15-5	619.38	932	618.89	930	615.20	880
15-6	1070.78	926	1070.10	895	1064.56	853
15-7	830.30	1034	823.89	1019	825.09	942
15-8	1023.44	1067	1017.39	1051	1012.42	985
15-9	687.50	1068	686.11	1077	678.51	1006
15-10	726.22	1017	721.71	998	723.27	957

overcome the shortcomings of the purely simulated annealing approach developed by Henderson *et al.* Experiments were conducted for comparison. For small problem instances, both algorithms are comparable providing near-optimal solutions. However, the difference in performance of the algorithms became clearer as problem size increased. For these problems ($n=15$), the TSSA was significantly faster than Henderson *et al*'s algorithm, with an improvement on average time of almost 55%. It was also more robust with an average standard deviation of 3.44 compared with 5.02 for Henderson *et al*'s algorithm. Finally, the average of the differences from the minimums obtained by the TSSA algorithms improved on those found by Henderson *et al*'s algorithm significantly.

Extensions of the problem were developed here. A different objective function is used to minimize the make-span. Experimental results show solutions to be comparable to the original ones. The problem is also extended to include multiple vehicles, which can cater to more realistic situations.

Table 4 Minimizing $C_{\max}$ *versus* minimizing $\sum R_i$

	$v=2$			$v=4$		
	Makespan		*Original*	*Makespan*		*Original*
Problem	C_{max}	$\sum R_i$	$\sum R_i$	C_{max}	$\sum R_i$	$\sum R_i$
5-1	25.07	50.12	48.13	13.30	49.53	48.13
5-2	39.97	79.66	76.70	20.60	80.13	76.29
5-3	28.24	56.37	53.79	14.24	55.02	52.55
5-4	90.57	180.28	179.06	46.26	180.36	178.26
5-5	55.54	110.99	110.16	28.85	110.62	109.89
5-6	33.05	65.37	64.97	18.00	65.45	64.97
5-7	30.19	59.73	57.51	15.90	59.10	56.51
5-8	63.34	125.49	124.52	32.17	126.67	124.48
5-9	44.91	89.36	89.07	23.59	91.07	89.00
5-10	72.99	145.84	144.36	37.42	146.26	144.25
7-1	135.42	270.64	267.82	69.23	275.44	267.82
7-2	53.83	107.62	103.45	27.74	105.71	103.05
7-3	70.87	141.23	137.31	36.29	142.88	136.55
7-4	131.35	262.62	257.94	66.71	266.13	257.78
7-5	54.90	109.35	106.35	28.77	113.09	104.94
7-6	65.64	130.24	127.10	35.00	137.16	126.90
7-7	77.67	154.32	150.09	40.78	159.49	148.31
7-8	138.46	276.32	271.29	69.97	278.22	270.71
7-9	67.13	132.97	128.61	35.53	134.75	126.97
7-10	57.51	114.84	109.73	30.54	114.70	109.73
9-1	106.11	212.19	202.18	55.08	217.95	201.83
9-2	301.21	602.29	595.59	152.74	610.61	595.99
9-3	148.54	295.86	286.22	77.27	307.45	286.22
9-4	121.52	242.64	234.33	63.35	251.02	234.50
9-5	123.74	247.21	240.04	64.62	256.46	239.26
9-6	129.93	259.79	248.97	67.74	268.69	249.01
9-7	226.19	452.27	446.67	114.73	457.95	445.71
9-8	116.78	232.46	221.80	60.31	240.44	222.67
9-9	153.96	307.18	300.32	78.24	312.01	300.08
9-10	188.49	376.36	369.05	95.68	382.10	369.25
15-1	401.47	802.64	764.07	211.54	842.80	759.66
15-2	339.11	678.10	646.72	179.50	712.53	640.28
15-3	596.76	1191.60	1169.28	306.48	1224.40	1164.13
15-4	479.54	956.99	928.50	249.46	993.13	924.74
15-5	332.61	663.70	618.89	176.35	704.71	615.20
15-6	548.68	1096.88	1070.10	281.48	1124.78	1064.56
15-7	437.68	875.05	823.89	227.43	906.02	825.09
15-8	527.44	1054.82	1017.39	273.45	1091.52	1012.42
15-9	362.54	724.03	686.11	190.26	759.83	678.51
15-10	385.02	768.17	721.71	203.95	806.65	723.27

References

1 Henderson D *et al* (2003). Solving the shortest route cut and fill problem using simulated annealing. *Eur J Opl Res* **145**: 72–84.

2 Laporte G (1992). The traveling salesman problem: an overview of exact and approximate algorithms. *Eur J Opl Res* **59**: 231–247.

3 Laporte G, Gendreau M, Potvin J-Y and Semet F (2000). Classical and modern heuristics for the vehicle routing problem. *Int Trans Opl Res* **7**: 285–300.

4 Charikar M, Khuller S and Raghavachari B (2001). Algorithms for capacitated vehicle routing. *SIAM J Comput* **31**: 665–682.

5 Anily S and Bramel J (1999). Approximation algorithms for the capacitated traveling salesman problem with pickups and deliveries. *Naval Res Logist* **46**: 654–670.

6 Renaud J, Boctor FF and Ouenniche J (2000). A heuristic for the pickup and delivery traveling salesman problem. *Comput Opns Res* **27**: 905–916.

7 Miller CE, Tucker AW and Zemlin RA (1960). Integer programming formulations and the traveling salesman problem. *J Assoc Comput Mach* **7**: 326–329.

8 Held M and Karp RM (1970). The traveling salesman problem and minimum spanning trees. *Opns Res* **18**: 1138–1162.
9 Held M and Karp RM (1971). The traveling salesman problem and minimum spanning trees: Part II. *Math Programm* **1**: 6–25.
10 Kirkpatrick S, Gelatt Jr CD and Vecchi MP (1983). Optimization by simulated annealing. *Science* **220**: 671–680.
11 Faigle U and Kern W (1992). Some convergence results for probabilistic tabu search. *ORSA J Comput* **4**: 32–37.
12 Glover F (1986). Future paths for integer programming and links to artificial intelligence. *Comput Opns Res* **13**: 533–549.
13 Moscato P (1993). An introduction to population approaches for optimization and heirachical objective functions: a discussion of the role of tabu search. *Ann Opns Res* **41**: 85–121.
14 Fleisher MA and Jacobsen SH (1999). Information theory and finite-time behavior of the simulated annealing algorithm: experimental results. *INFORMS J Comput* **11**: 35–43.

Received November 2003;
accepted September 2004 after two revisions