Available online at www.sciencedirect.com
SCIENCE DIRECT®
Transportation Research Part E 41 (2005) 115–130
TRANSPORTATION RESEARCH PART E
www.elsevier.com/locate/tre

Meta-RaPS: a simple and effective approach for solving the traveling salesman problem

Gail W. DePuy [a,*], Reinaldo J. Moraga [b], Gary E. Whitehouse [c]

[a] *Department of Industrial Engineering, University of Louisville, Louisville, KY 40292, USA*
[b] *Department of Industrial Engineering, Universidad del Bío Bío Avenida Collao 1202, Casilla 5C, Concepción, Chile*
[c] *Industrial Engineering and Management Systems Department, University of Central Florida, Orlando, FL 32816-2450, USA*

Received 7 October 2003; received in revised form 4 February 2004; accepted 10 February 2004

Abstract

This paper investigates the development and application of a general meta-heuristic, Meta-RaPS (meta-heuristic for randomized priority search), to the traveling salesman problem (TSP). The Meta-RaPS approach is tested on several established test sets. The Meta-RaPS approach outperformed most other solution methodologies in terms of percent difference from optimal. Additionally, an industry case study that incorporates Meta-RaPS TSP in a large truck route assignment model is presented. The company estimates a more than 50% reduction in engineering time and over $2.5 million annual savings in transportation costs using the automated Meta-RaPS TSP tool compared to their current method.
© 2004 Elsevier Ltd. All rights reserved.

Keywords: Meta-heuristic; TSP; Cheapest insertion

1. Introduction

The well-known traveling salesman problem (TSP), in its simplest form, involves finding an optimal route for visiting N cities and returning to the point of origin, where the inter-city distances are symmetric and known. Regardless of the fact that TSP can be easily formulated, its nature exhibits all aspects of NP-hard combinatorial optimization (Reinelt, 1994). This is the

* Corresponding author. Tel.: +1-502-852-0115; fax: +1-502-852-5633.
E-mail addresses: depuy@louisville.edu (G.W. DePuy), rmoraga@ubiobio.cl (R.J. Moraga), whitehse@mail.ucf.edu (G.E. Whitehouse).

1366-5545/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.tre.2004.02.001

reason why TSP has been intensively studied, because any improvement in finding TSP optimized solutions can be adapted to the entire class of NP-hard problems (Johnson, 1991). This paper investigates the development and application of a general meta-heuristic, Meta-RaPS (meta-heuristic for randomized priority search), to the traveling salesman problem.

Past research for the TSP has been extensive. Several good TSP survey papers are available. Melamed et al. (1990) provide various descriptions of the TSP and its formulations are examined in linear and integer linear programming forms. Burkard et al. (1998) review several polynomial time TSP heuristics developed during the decade 1985–1995. Also, Moraga and AlMazid (2000) provide a survey of the TSP and its industrial applications found in the literature from 1980 to 1999. Laporte et al. (1996) review a wide variety of different combinatorial optimization problems that can be formulated as a generalized TSP. Finally, Reinelt (1994), in his text, provides an excellent treatment to the TSP subject, covering its computational solutions and applications.

While the goal of some TSP research is to find an algorithm that gives an optimal solution in polynomial time with respect to the problem size, the main interest in practice is to guarantee a good quality solution (or near-optimal solution) in a reasonable amount of time. Numerous approaches to solve TSP optimization problems have been proposed, varying from brute-force enumeration through highly esoteric optimization methods. The majority of these methods can be broadly classified as either "exact" algorithms or "heuristic" algorithms. Exact algorithms are those that yield an optimal solution. The branch-and-bound method, for example, is a quasi-enumerative approach that has been used in a wide variety of combinatorial problems. Branch-and-bound is reasonably efficient for problems of modest size and, as a general methodology, it constitutes an important part of the set of exact solution methods for the general class of integer linear programming problems (Ingnizio and Cavalier, 1994). Recent TSP studies using the branch and bound approach are Lysgaard (1999) and Fischetti and Toth (1997). The size of practical problems frequently precludes the application of exact methods and thus heuristic algorithms are often used for real-time solution of the TSP.

Much research has been done on heuristic algorithms for the TSP. Johnson and McGeoch (2002) provide a discussion of several TSP heuristics. For the purposes of this paper, TSP heuristics will be classified as either classic construction TSP heuristics or modern TSP heuristics. Several well-know classic construction TSP heuristics include nearest neighbor (see Reinelt, 1994), insertion heuristics (see Reinelt, 1994), convex hull (see Lawler et al., 1985), Delaunay Triangulation (see Krasnogor et al., 1995), minimum spanning tree (Christofides, 1976), and savings methods (Clarke and Wright, 1964). Every TSP construction heuristic has, in theory, the chance to find an optimal solution. But, finding the optimal solution typically is a remote event because heuristics often get stuck in a local optimal solution. Modern heuristics deal with this problem by introducing systematic rules to move out of a local minimum or avoid a local minimum.

Modern TSP heuristics approaches have appeared during the last two decades. They take an initial solution, which is improved by using improvement heuristics embedded in a more general framework. The common characteristic of these modern heuristics is the use of randomness as a mechanism to avoid local optima. Modern heuristics such as genetic algorithms (GA), simulated annealing (SA), tabu search (TS), and neural networks (NN) succeed in leaving the local optimum by temporarily accepting moves which cause a worsening of the objective function value. For example, in the case of simulating annealing a bad solution can be accepted as appropriate with a certain probability. In addition to the idea of escaping from local optima, genetic algorithms use randomness to generate an initial population of solutions before starting the search. However, when the problem is too unconstrained, as the case of large TSPs, a fully random selection of the initial population is not helpful. Some authors recommend the addition of problem knowledge to the algorithm through choosing the initial population according to a heuristic (Chatterjee et al., 1996).

The following articles report the use of modern heuristics for solving the TSP: genetic algorithms (Chatterjee et al., 1996; Schmitt and Amini, 1998); neural network (Altinel et al., 2000; Burke, 1996; Burke and Damany, 1992; Matsuyama, 1992; Modares et al., 1999; Soylu et al., 2000); simulated annealing (Malek et al., 1989; Yip and Pao, 1995) and tabu search (Gendreau et al., 1998; Malek et al., 1989; Tsubakitani and Evans, 1998). These heuristics are often called "meta-heuristics" because the procedure used to generate a new solution out of the current one is embedded in a heuristic that determines the search strategy. This paper details the application of another meta-heuristic, Meta-RaPS, to the traveling salesman problem.

Meta-RaPS (meta-heuristic for randomized priority search) is a generic, high level strategy used to modify construction heuristics based on the insertion of randomness. Meta-RaPS integrates priority rules, randomness, and sampling. At each iteration, Meta-RaPS constructs and improves feasible solutions through the utilization of construction heuristic priority rules used in a randomized fashion. After a number of iterations, Meta-RaPS reports the best solution found. As with other meta-heuristics, the randomness represents a device to avoid getting stuck in a local optima.

In general, a construction heuristic builds a solution by repeatedly adding feasible components or activities to the current solution until a stopping criteria is met. The order in which the activities are added to the solution is based on their priority values as determined by the construction heuristic's priority rule. The priority value of each feasible activity is calculated and the activity with the best or most desirable priority value is added to the current solution next. For example, a TSP construction heuristic builds a solution by repeatedly adding unvisited cities (i.e. feasible activities) to the route (i.e. solution) until all the cities are included in the route (i.e. stopping criteria). In most TSP construction heuristics the order in which the cities are added to the route is based on minimizing the total route distance (i.e. priority rule) by calculating the increase in route distance (i.e. priority value) for each feasible city and inserting the city with the minimum increase in route distance (i.e. best priority value). Solutions generated using a construction heuristic are usually good, but rarely optimal, as the construction heuristic often generates a local optimum solution. Using Meta-RaPS to include randomness in a construction heuristic frequently allows the global optimum solution to be found as will be demonstrated later in this paper.

Meta-RaPS modifies the way a general construction heuristic chooses the next activity to add to the solution by occasionally choosing an activity that does not have the best priority value. Sometimes the next activity is chosen randomly from those feasible activities with good, but not the best, priority values. Meta-RaPS uses two user-defined parameters, % priority and % restriction, to include randomness in a construction heuristic (see Fig. 1). The % priority parameter is used to determine how often the next activity added to the solution has the best priority value. The remaining time (100%–% priority) the next activity added to the solution is randomly chosen from an 'available list' of those feasible activities with good priority values. Good priority values are determined using the % restriction parameter. Those feasible activities

```
Do Until feasible solution generated
    Find priority value for each feasible activity
    Find best priority value
    P = RND(1,100)
    If P<= %priority Then
        Add activity with best priority value to solution
    Else
        Form 'available list' of all feasible activities whose priority values are within %restriction
              of best priority value
        Randomly choose activity from available list and add to solution
    End If
End Until
Calculate and Print solution value
```

Fig. 1. Pseudocode for one iteration of basic Meta-RaPS procedure.

whose priority values are within % restriction of the best priority value are included on the 'available list.' Again, an activity is then randomly selected from this list to be added to the solution next. Therefore, each application or iteration of the heuristic will most likely produce a different solution because the same activity will not be randomly selected from the available list each time the heuristic is used. An appropriate methodology to select the % priority and % restriction parameter values is discussed in Section 2.2.

In addition, a solution improvement algorithm can be included in Meta-RaPS using a % improvement parameter. If an iteration's solution value is within % improvement of the best unimproved solution value found so far, then an improvement heuristic (neighborhood search) is performed. As mentioned previously, Meta-RaPS is a general solution approach that can be applied to a variety of combinatorial optimization problems including the TSP.

An introduction to the general Meta-RaPS approach as well as a description of the Meta-RaPS procedure as applied to the TSP are presented in the next section. To demonstrate the solution quality obtainable with Meta-RaPS TSP, the Meta-RaPS approach is tested on several established test sets as presented in Section 3. Solution values are compared to both optimal values and the results of other TSP solution techniques. Because the Meta-RaPS approach is easy to understand and implement and because Meta-RaPS TSP achieves good results, it should be an attractive solution methodology for industry practitioners. Section 4 details an industry case study that incorporates Meta-RaPS TSP in a large truck route assignment model developed by a well-known national trucking company.

2. Meta-RaPS approach for solving the TSP

2.1. General Meta-RaPS approach

Meta-RaPS is the result of research conducted on the application of a modified COMSOAL approach to several combinatorial problems. COMSOAL (computer method of sequencing operations for assembly lines) is a computer heuristic originally reported as a solution approach to the assembly line balancing problem (Arcus, 1966). However, the underlying concept of this heuristic can be applied to a variety of problems. Despite the fact that each modified version of COMSOAL discussed by DePuy and Whitehouse (2000, 2001) conserves in essence Arcus's underlying idea, in practice, the versions differ considerably from the original COMSOAL. This led DePuy and Whitehouse to present their approach under the name of "Meta-RaPS".

For a general combinatorial problem, COMSOAL constructs solutions by generating a list of those feasible activities that are available to be included in the current solution. The next activity to be added to the solution is randomly chosen from this 'available list.' Meta-RaPS modifies the basic COMSOAL approach by either randomly choosing the next feasible activity from the available list or selecting the activity with the best priority value. A user-defined distribution (% priority) is employed to determine whether the next activity is chosen at random or by using a priority scheme. Meta-RaPS further modifies the original COMSOAL approach by limiting the activities that appear in the available list to those whose priority values are close to the best priority value through the use of the % restriction parameter. Finally Meta-RaPS differs from COMSOAL in that it includes the option of employing an improvement heuristic. Based on a user-defined distribution, a percentage of the feasible solutions constructed by Meta-RaPS can be improved using a problem specific or general neighborhood search improvement heuristic.

Meta-RaPS can be considered a general form of COMSOAL. Using Meta-RaPS parameters of 0% priority, an infinitely large % restriction, and 0% improvement will mimic COMSOAL. Meta-RaPS is also the general form of another meta-heuristic called GRASP (Greedy Randomized Adaptive Search Procedure; Feo and Resende, 1989). GRASP constructs solutions by introducing randomness to a greedy construction heuristic through the use of a % restriction parameter in the same way as Meta-RaPS. GRASP also includes a local search improvement heuristic applied to all constructed solutions. GRASP has been applied to a variety of problems including machine scheduling (Rios-Mercado and Bard, 1998; Feo et al., 1996), quadratic assignment problem (Mavridou et al., 1998), aircraft routing (Arguello et al., 1997) and fleet scheduling (Sosnowska, 2000), however no research solving TSP problems using GRASP has been found to date. Using Meta-RaPS parameters of 0% priority and 100% improvement will imitate GRASP. Meta-RaPS offers greater flexibility over COMSOAL and GRASP in that it allows user-defined settings for three parameters (% priority, % restriction, and % improvement). In addition, GRASP uses greedy algorithms while Meta-RaPS has been used with both greedy algorithms and non-greedy or look-ahead priority rules (DePuy and Whitehouse, 2000, 2001). As will be demonstrated in the following sections, the Meta-RaPS' parameter settings used to find the best solutions are settings other than those that imitate COMSOAL or GRASP. Therefore, the extra flexibility afforded by the general form of Meta-RaPS seems beneficial.

2.2. Meta-RaPS TSP

This paper will consider the use of two classic TSP construction heuristics, nearest neighbor and cheapest insertion, as priority rules within Meta-RaPS. The nearest neighbor, NN (see Reinelt, 1994) heuristic starts at a randomly chosen initial city. The travel route is built one city at a time, with the next city in the route always being that unvisited city that is closest to the current city. The cheapest insertion, CI (see Reinelt, 1994) heuristic starts with two randomly chosen cities. The method then finds the best insertion (i.e., the insertion into the current route that minimizes the total distance) for each unvisited city. The overall best/cheapest insertion is then made. Unvisited cities are evaluated in this manner until all cities have been placed in the tour. Both heuristics select from all unvisited cities that city whose addition to the route causes the smallest increase in the length of the tour (Reinelt, 1994). These two simple TSP heuristics can be modified using Meta-RaPS to incorporate randomness.

In Meta-RaPS TSP, the % priority parameter defines the percentage of time the next city added to the route is the city with the best priority value as defined by the simple heuristic (NN or CI). The remaining time (i.e. 100−% priority) the next city added to the route is randomly chosen from those cities whose priority values are within % restriction of the best priority value. For example, the Meta-RaPS nearest neighbor approach sometimes (% priority of the time) selects the nearest

```
iteration =1
Do While iteration <= num_iterations
    'begin tour construction phase'
    randomly select two initial cities
    form tour
    numcities_in_tour=2
    If numcities_in_tour < num_cities Then
        Find overall_cheapest_city and associated overall_cheapest_tour_position
        Find overall_cheapest_tour_distance_increase
        P = RND(1,100)
        If P<= %priority Then
            'do overall cheapest insertion'
            Insert overall_cheapest_city in overall_cheapest_tour_position
        Else
            'do insertion of available list city'
                'form available list'
                For each unvisited city
                    Find cheapest_tour_position and associated cheapest_tour_distance_increase
                    If cheapest_tour_distance_increase <= 0.01*(100+%restriction)*
                        overall_cheapest_tour_distance_increase Then Add city to available list
                Next unvisited city
                Randomly choose city from available list
                Insert chosen available list city in tour in its cheapest_tour_position
        End If
        numcities_in_tour= numcities_in_tour + 1
        'end of tour construction phase'
    Else
        Calculate tour_distance
        If tour_distance < best_tour_distance Then best_tour_distance= tour_distance
        If tour_distance <= 0.01*(100+%improve)*best_tour_distance Then
            Do improvement
            Calculate tour_distance
        End If
        If tour_distance < best_final_distance Then best_final_distance= tour_distance
        iteration=iteration+1
End While
Print best_final_distance
```

Fig. 2. Pseudocode of Meta-RaPS cheapest insertion TSP procedure.

Table 1
Meta-RaPS parameter settings used

	% Priority	% Restriction
Meta-RaPS NN	80	30
Meta-RaPS CI	20	10

neighbor (i.e. best priority value city) as the next scheduled city. The remaining time, a list of those unvisited cities whose distances are within % restriction of the nearest neighbor is formed and a city is randomly selected from the list for inclusion in the current tour. Specifically, if parameters of 20% priority and 40% restriction are used, then 20% of the time the next scheduled city is the nearest city and the remaining 80% of the time the next scheduled city is randomly selected from a list of those unvisited cities that are within a distance of 1.40* (distance to the nearest city). This method is thought to decrease the possibility of arriving at solutions that are caught in local optima, and helps the user to find an answer closer to the global optimum. The Meta-RaPS TSP approach using cheapest insertion as the priority rule is outlined in the Fig. 2 pseudocode.

When applying the Meta-RaPS method, several parameters must be set. Coy et al. (2001) note that it is often difficult to find appropriate parameter settings for meta-heuristics and common procedures have ranged from simple trial-and-error to complicated sensitivity analysis. A brief trial and error procedure was used in this research. Two representative TSP problems were selected from the TSPLIB (Reinelt and Bixby, 1995), a set of TSP test problems commonly used to benchmark TSP solution methodologies. These two representative problems are a subset of the 22 Euclidean TSPLIB problems used throughout this paper. Based on previous Meta-RaPS TSP experience (Schulman, 2001), the following parameters settings were considered: % priority (10%, 20%, 70%, 80%) and % restriction (10%, 20%, 30%). One thousand iterations were run at each of the 12 parameter combinations for the two representative TSP problems using both Meta-RaPS nearest neighbor and Meta-RaPS cheapest insertion. Based on this simple parameter search, the parameter combinations shown in Table 1 were chosen. The same parameter settings (i.e. those shown in Table 1) were then used for all 22 TSPLIB test problems solved throughout this paper.

3. Meta-RaPS results

This results section is presented in three parts. First the benefit of incorporating randomness in a simple priority rule (i.e. the basic premise of the Meta-RaPS approach) is investigated by solving several test problems with and without randomness. Next the benefit of including an improvement heuristic in the Meta-RaPS approach is discussed. Finally, the solution quality of Meta-RaPS will be compared to that of several other TSP solution techniques through the use of benchmark TSP test problems.

3.1. Results of basic Meta-RaPS approach

The Meta-RaPS approach described in Section 2 was coded in C++ and used to solve nine problems from the TSPLIB. These nine test problems were selected because other researchers

Table 2
Percent difference from optimal for 5000 iterations of each problem using each of four solution methods

Problem	# Cities	% Difference from optimal			
		NN	Meta-RaPS NN	CI	Meta-RaPS CI
Mpr21	21	10.93	7.57	7.06	0.00
Mgr48	48	15.74	6.36	3.79	0.02
Mpr76	76	21.04	12.88	4.91	1.00
KroA	100	16.05	11.30	7.31	0.50
KroB	100	16.90	6.34	7.41	0.97
KroC	100	13.58	6.45	9.36	0.85
KroD	100	16.73	9.39	5.36	1.45
KroE	100	12.86	10.76	4.99	1.30
Meil101	101	17.07	13.57	9.86	5.23
Average		15.66	9.40	6.67	1.26

have reported results for these same problems and therefore a basis for comparison is provided. The selected test problems from TSPLIB were solved using the nearest neighbor, Meta-RaPS nearest neighbor, cheapest insertion, and Meta-RaPS cheapest insertion methods. The results are shown in Table 2. Each problem was run for 5000 iterations using each of four solution methods on an AMD Athlon microprocessor PC and the best solution value is reported. Run times for 5000 iterations of the 100-city problems using the Meta-RaPS cheapest insertion heuristic were approximately 22 s. As can be seen in Table 2, the inclusion of randomness (i.e. the Meta-RaPS approach) always outperformed the priority rule. It can also be seen that the cheapest insertion priority rule dominates the nearest neighbor priority rule. Based on these findings, only the Meta-RaPS cheapest insertion approach will be used throughout the remainder of this paper.

3.2. Meta-RaPS approach with improvement heuristic

Now that the benefits of including randomness in a simple construction priority rule have been demonstrated, the task at hand becomes solution improvement. The node insertion heuristic (see Reinelt, 1994) can be included in the Meta-RaPS cheapest insertion heuristic to improve solutions. In addition to the two parameters previously discussed (% priority and % restriction), Meta-RaPS will now use the parameter % improvement for choosing whether or not to improve the tour. If a tour's solution value before improvement is within % improvement of the best unimproved solution value found in a previous iteration, the node insertion improvement heuristic will be performed. The idea behind this strategy is that good unimproved solutions lead to good improved solutions. Fig. 3 and Table 3 support this premise. Fig. 3 shows the relationship between the unimproved and corresponding improved solution values for the Mpr76 problem. For each of 200 iterations the solution value before and after improvement is compared to the optimal solution value and the percent difference from optimal is shown in Fig. 3. As demonstrated in Fig. 3, those iterations with good solution values before improvement (i.e. small percent difference from optimal) led to very good solutions after the improvement heuristic was employed. While it is quite possible an unimproved solution of moderate quality could lead to a good improved solution, the best improved solutions come from good unimproved solutions.

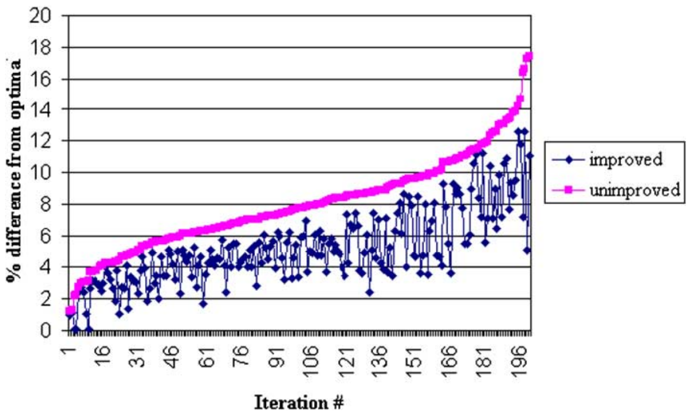

Fig. 3. Relationship between unimproved and improved solution values for 200 iterations of the Mpr76 (76 city) problem.

Table 3
Correlation between unimproved and improved solutions

Problem	# Cities	Correlation between unimproved and improved % difference from optimal solution values
Mpr21	21	0.68
Mgr48	48	0.75
Mpr76	76	0.80
KroA	100	0.78
KroB	100	0.72
KroC	100	0.75
KroD	100	0.78
KroE	100	0.72
Meil101	101	0.55

Table 3 shows the strong positive correlation between the unimproved and improved solution values measured in terms of percent difference from optimal solution value. For each Table 3 problem, 200 iterations were conducted and the percent different from the optimal solution value was recorded for both the before and after improvement solutions for each iteration. The correlation coefficient was then calculated for each problem and reported in Table 3. The Table 3 correlation values being near 1 is an indication that good improved solutions came from good unimproved solutions. Based on these results, Meta-RaPS will allow the users to concentrate on improving those solutions that show the most potential for a good, final solution value.

3.3. Meta-RaPS results—A comparison of solution quality to other TSP techniques

The Meta-RaPS cheapest insertion heuristic using 20% priority, 10% restriction, 5% improvement, and 10,000 iterations was used to solve five 100-city problems. These five test problems were chosen as they have been used by many researchers and therefore offer a

Table 4
Percent difference from optimal for five TSPLIB test problems using various solution methodologies

Solution methodology	Problem				
	KroA	KroB	KroC	KroD	KroE
Meta-RaPS TSP	0.00%	0.25%	0.00%	0.00%	0.17%
Priority rule (CI & node insertion)	0.50%	2.46%	0.82%	1.43%	1.10%
GRASP	0.00%	0.55%	0.31%	0.42%	0.37%
Christofides & 2opt 3opt (Lawler et al., 1985)	2.51%	1.40%	1.53%	0.17%	3.03%
Convex hull & 3opt (Lawler et al., 1985)	0.37%	1.46%	1.06%	0.04%	2.46%
Nearest neighbor & 2opt 3opt (Lawler et al., 1985)	0.14%	1.46%	1.06%	0.73%	2.46%
Lin Kernihan (Padberg and Rinaldi, 1991)	0.26%	**0.00%**	0.70%	0.17%	**0.16%**
Modified Lin Kernihan (Mak and Morton, 1992)	0.00%	**0.17%**	0.00%	0.00%	0.21%
Composite heuristic, CCAO (Golden and Stewart, 1985)	0.00%	0.97%	0.50%	0.97%	2.54%
Delaunay triangulation heuristic (Krasnogor et al., 1995)	0.51%	2.13%	2.79%	3.81%	2.00%
I^3 heuristic (Renaud et al., 1996)	0.00%	0.90%	0.50%	2.00%	2.60%
P-SEC algorithm (Rego, 1998)	0.00%	0.32%	0.02%	0.75%	0.33%
F-SEC algorithm (Rego, 1998)	0.00%	**0.00%**	0.00%	0.00%	**0.00%**
Guided local search (Voudouris and Tsang, 1999)	0.00%	N/A	0.00%	N/A	N/A
Genetic algorithm, A (Chatterjee et al., 1996)	0.70%	N/A	1.78%	1.45%	N/A
Genetic algorithm, B (Chatterjee et al., 1996)	1.80%	N/A	2.10%	1.30%	N/A
Neural net (Modares et al., 1999)	0.31%	1.43%	N/A	N/A	N/A
Neural net (Matsuyama, 1992)[a]	1.81%	3.37%	N/A	N/A	N/A
Neural net (Burke and Damany, 1992)[a]	5.00%	4.31%	N/A	N/A	N/A
Simulated annealing (Voudouris and Tsang, 1999)	0.42%	N/A	0.80%	N/A	N/A
Simulated annealing (Malek et al., 1989)	0.09%	N/A	N/A	N/A	N/A
Tabu search (Voudouris and Tsang, 1999)	0.00%	N/A	0.25%	N/A	N/A
Tabu search (Malek et al., 1989)	0.33%	N/A	N/A	N/A	N/A
Tabu search & 3 opt (Tsubakitani and Evans, 1998)	1.37%	N/A	N/A	N/A	N/A

[a] As reported by (Modares et al., 1999).

comparison of solution values. Table 4 shows the solution quality, expressed as percent difference from optimal, of Meta-RaPS compared to various other TSP solution methodologies.

As can be seen in Table 4, the solution quality of the Meta-RaPS approach compares favorably to the vast majority of the other methods. The bold numerals in Table 4 show those instances where Meta-RaPS was outperformed. Table 4 shows Meta-RaPS outperforms the other meta-heuristics such as GRASP, neural networks, simulated annealing, tabu search, and genetic algorithms for these selected problems. It should be noted that no published research could be found using GRASP to solve a TSP, therefore the GRASP results shown in Table 4 were

Table 5
Sample of computer platforms and run times for 100 city KroA problem

Solution methodology	Computer platform	Programming language	Run time
Meta-RaPS TSP	AMD 900 MHz Athlon PC	C++	50 s
GRASP	AMD 900 MHz Athlon PC	C++	8.5 min
I^3 heuristic (Renaud et al., 1996)	Sun Sparcstation 2	Pascal	2 s
Guided local search (Voudouris and Tsang, 1999)	DEC Alpha 3000	N/A	1.3 s
Genetic algorithm (Chatterjee et al., 1996)	Vax 8600	Pascal	1 h
Neural net (Modares et al., 1999)	SunSparcstation 10	C	55 s
Simulated annealing (Voudouris and Tsang, 1999)	DEC Alpha 3000	N/A	37 s
Tabu search (Voudouris and Tsang, 1999)	DEC Alpha 3000	N/A	21 s
Tabu Search & 3 opt (Tsubakitani and Evans, 1998)	386 microcomputer	C	2 h

generated by the authors using the cheapest insertion construction heuristic (10% restriction) and the node insertion improvement heuristic (100% improvement).

Although the main focus of this paper is the solution quality obtained using Meta-RaPS, some mention must be made regarding run times. It is often difficult to compare the run times of different solution techniques as different computer platforms, operating systems, and programming languages are used by researchers. A sample of the run times associated with the Table 4 results is shown in Table 5. As mentioned before, Meta-RaPS should be an attractive solution technique for industry practitioners since it is simple in concept, easy to implement, and achieves good results. Therefore, a computer system and programming language that are familiar to most practitioners, an AMD Athlon microprocessor PC and C++, were used throughout this research. Run times for 10,000 iterations of the 100-city KroA problem using 20% priority, 10% restriction, and 5% improvement were approximately 50 s.

Table 6
Percent difference from optimal or best known for TSPLIB test problems using Meta-RaPS TSP

Problem	# Cities	# Iterations	Run time (in s)	% Difference from optimal or best known
lin105	105	5000	20	0.00
pr107	107	5000	139	0.00
pr124	124	5000	49	0.00
bier127	127	5000	48	0.90
pr136	136	5000	73	0.39
pr152	152	5000	178	0.00
KroA200	200	2000	190	1.07
KroB200	200	2000	134	1.26
pr226	226	2000	357	0.23
pr264	264	2000	824	1.58
pr299	299	2000	766	2.01
pr439	439	2000	2265	3.29
pr1002	1002	1000	7032	6.04

While the five 100-city test problems shown in Table 4 have become a standard for comparison, there are other TSP test problems available, although none have been featured in the literature as often as the five 100-city test problems shown in Table 4. Table 6 shows the Meta-RaPS TSP results for several, larger TSP problems found in TSPLIB. These problems were run using parameters of 20% priority, 10% restriction, and 5% improvement. The number of iterations and computer run times are shown in Table 6. Meta-RaPS TSP, as with most other TSP solution methodologies, seems to decline in solution quality as the problem size increases.

4. Case study application

The Meta-RaPS TSP approach developed in this paper was incorporated in a large truck route assignment model developed by a well-known national trucking company. Our industrial partner developed a model to generate routes for their less than truckload (LTL) pick-up and delivery operations. These operations entail the morning and early afternoon delivery of freight from a terminal to customers as well as the late afternoon pick-up of freight from customers. Multiple pick-up and delivery routes around each terminal must be generated as the number of customers is more than can be serviced by one truck. These routes are generated with a goal of evenly distributing the workload among trucks/routes. Therefore the company wishes to generate multiple pick-up and delivery routes around each terminal such that each route contains 6.5 h of deliveries. With over 300 terminals nationally and each terminal requiring multiple pick-up and delivery routes, many potential routes have to be generated and evaluated. Therefore an automated system to generate pick-up and delivery routes around a terminal was developed by the company. Within their automated system, Meta-RaPS TSP is used to solve the TSP and generate a specific route for subsets of customers.

The iterative route generating procedure developed by our industrial partner can be described as follows. They first create a circular zone around a terminal such that the driver has approximately 6.5 h of work. Then they construct zones outside the inner circle by sweeping around the

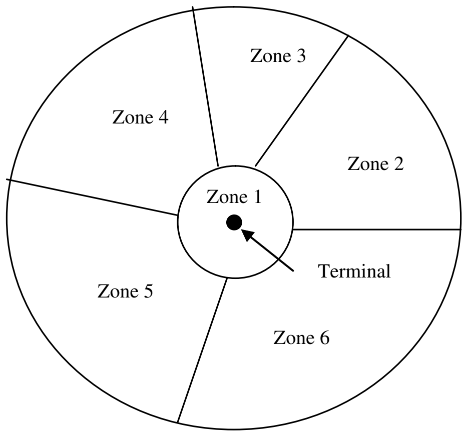

Fig. 4. Typical route zone pattern for a terminal.

inner circle. Each of the outer zones has approximately 6.5 h of delivery allotted to it. A typical pattern of route zones is shown in Fig. 4. Having an inner circle or zone and outside zones places the drivers in better geographic positions to make afternoon pickups after their deliveries are finished. Within each of these zones, Meta-RaPS TSP is used to generate the specific route and therefore estimate the delivery time associated with each zone. The sizes of the outer zones (i.e. zones 2, 3, ...) are adjusted such that each outer zone contains 6.5 h of deliveries. The workload within zone 1 is regarded as slack and zone 1 is also adjusted in size so as to balance the workload load among the outer zones. Customers are shifted in and out of zone 1 so as to balance the outer zones to approximately 6.5 h of deliveries. This zone size adjustment process is iterative in nature and requires multiple TSP problems to be solved at each iteration.

For a problem size of 200 customers per terminal, the iterative route generating procedure described above is solved in approximately 30 min using Meta-RaPS TSP coded in Visual Basic running on a 766 MHz laptop PC. Meta-RaPS TSP parameters of 20% priority, 10% restriction, and 2000 iterations were used. The company's previous method of route assignment involved using industrial engineers to manually generate the routes for each terminal. This manual procedure required one week of engineering time per terminal. Therefore, with over 300 terminals, a team of six engineers would require one year to manually generate routes for all terminals. Using the automated system, our industrial partner estimates an engineering time savings of at least 50%. In addition to the savings in engineering time, the automated route system is thought to produce shorter routes since Meta-RaPS TSP has been shown to generate high quality, near optimal routes. If even the relatively small savings of 30 miles per day per terminal is realized, the company has estimated the annual financial savings due to mileage reduction to be in excess of $2.5 million.

Based on the promising results of applying Meta-RaPS to the TSP, our industrial partner is considering employing Meta-RaPS to solve several other freight logistics problems such as redistribution of vehicles to terminals, assignment of freight to trailers, determination of number and location of terminals in metropolitan areas, and assignment of trailers to unloading and loading doors so as to minimize crossdock material handling.

5. Conclusions

The main objective of this research was to develop a simple method to find good solutions to traveling salesman problems. The Meta-RaPS approach was introduced as a method of incorporating randomness in established TSP priority schemes. The two TSP priority schemes considered in this paper, nearest neighbor and cheapest insertion, achieved significantly better results using Meta-RaPS' combination of randomness and priority schemes than using the priority scheme exclusively. In addition, improvement heuristics can be included in the Meta-RaPS approach to further improve promising solutions.

The Meta-RaPS method should be particularly attractive to industrial practitioners as it is easy to comprehend, easy to put into practice, and gets good results in a reasonable amount of time even for large problems. The Meta-RaPS TSP heuristic developed in this paper achieved good results when compared to both the optimal solution and other TSP heuristics. The advances of the Meta-RaPS TSP heuristic over other TSP heuristics include:

- Meta-RaPS TSP is easy to understand and easy to implement (i.e. write computer code). Many existing TSP methods are complex and require a mature understanding of mathematical programming and/or computer programming.
- Meta-RaPS TSP generates a feasible tour at every iteration. Therefore the method can be stopped at any time and a feasible solution is available. Some existing TSP methods could require much time to build a single solution for a large problem. Therefore, with some TSP methods even a feasible solution may not be able available after much run time.
- Meta-RaPS TSP produces results that are, in the majority of cases, better than the results of other TSP heuristics.

The use of Meta-RaPS for other combinatorial problems is now being considered. Future research is directed at applying the Meta-RaPS approach to other combinatorial problems such as the vehicle routing problem, bin packing problem, and machine scheduling problem.

References

Altinel, K., Aras, N., Oommen, J., 2000. Fast, efficient and accurate solutions to the hamiltonian path problem using neural approaches. Computers and Operations Research 27 (5), 461–494.

Arcus, A., 1966. COMSOAL: A computer method of sequencing operations for assembly lines. International Journal of Production Research 4, 259–277.

Arguello, M., Bard, J., Yu, G., 1997. A GRASP for aircraft routing in response to groundings and delays. Journal of Combinatorial Optimization 1, 211–228.

Burkard, R., Deineko, V., Van Dal, R., Van Der Veen, J., Woeginger, G., 1998. Well-solvable special cases of the traveling salesman problem: A survey. SIAM Review 40 (3), 496–546.

Burke, L., 1996. Conscientious neural nets for tour construction in the traveling salesman problem. Computers and Operations Research 23 (2), 121–129.

Burke, L.I., Damany, P., 1992. The guilty net for the traveling salesman problem. Computers and Operations Research 19 (4), 255–265.

Chatterjee, S., Carrera, C., Lynch, L., 1996. Genetic algorithms and traveling salesman problems. European Journal of Operational Research 93, 490–510.

Christofides, N., 1976. Worst-case analysis of a new heuristic for the traveling salesman problem. Management Sciences Research Report No. 388, Carnegie-Mellon University.

Clarke, G., Wright, J., 1964. Scheduling of vehicles from a central depot to a number of delivery points. Operations Research 12, 568–581.

Coy, S., Golden, B., Runger, G., Wasil, E., 2001. Using experimental design to find effective parameter setting for heuristics. Journal of Heuristics 7, 77–97.

DePuy, G., Whitehouse, G., 2001. A simple and effective heuristic for the resource constrained project scheduling problem. International Journal of Production Research 39 (14), 3275–3287.

DePuy, G., Whitehouse, G., 2000. Applying the COMSOAL computer heuristic to the constrained resource allocation problem. Computers and Industrial Engineering 38, 413–422.

Feo, T., Resende, M., 1989. A probabilistic heuristic for a computationally difficult set covering problem. OR Letters 8, 67–71.

Feo, T., Sarathy, K., McGahan, J., 1996. A GRASP for single machine scheduling with sequence dependent setup costs and linear delay penalties. Computers and Operations Research 23, 881–895.

Fischetti, M., Toth, P., 1997. A polyhedral approach to the asymmetric traveling salesman problem. Management Science 43 (11), 1520–1536.

Gendreau, M., Laporte, G., Semet, F., 1998. A tabu search heuristic for the undirected selective traveling salesman problem. European Journal of Operational Research 106 (2–3), 539–545.

Golden, B.L., Stewart, W.R., 1985. Empirical analysis of heuristic. In: Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., Shmoys, D.B. (Eds.), The Traveling Salesman Problem. John Wiley & Sons, New York, pp. 207–249.

Ingnizio, J., Cavalier, T., 1994. Linear Programming. Prentice Hall.

Johnson, D., 1991. The Traveling Salesman Problem: A Report on the State of the Art. In: Pehrson, B., Simon, I. (Eds.), 13th World Computer Congress, Elsevier Science, pp. 221–222.

Johnson, D.S., McGeoch, L.A., 2002. Experimental analysis of heuristics for STSP. In: Gutin, G., Punnen, (Eds.), The Traveling Salesman Problem and its Variations. Kluwer Academic Publishers, pp. 369–443.

Krasnogor, N., Moscato, P., Norman, M.G., 1995. A new hybrid heuristic for large geometric traveling salesman problems based on the delaunay triangulation. Anales del xxvii Simposio Brasileiro de Pesquisa Operacional.

Laporte, G., Asef-Vaziri, A., Sriskandarajah, C., 1996. Some applications of the generalized traveling salesman problem. Journal of The Operational Research Society 47, 1461–1467.

Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., Shmoys, D.B., 1985. The Traveling Salesman Problem. John Wiley & Sons, New York.

Lysgaard, J., 1999. Cluster based branching for the asymmetric traveling salesman problem. European Journal of Operational Research 119 (2), 314–325.

Mak, K., Morton, A., 1992. A modified Lin–Kernighan traveling salesman heuristic. ORSA Journal on Computing 13, 127–132.

Malek, M., Guruswamy, M., Pandya, M., Owens, H., 1989. Serial and parallel simulated annealing and tabu search algorithms for the traveling salesman problem. Annals of Operations Research 21, 59–84.

Matsuyama, Y., 1992. Self-organization neural networks and various euclidean traveling salesman problems. Systems and Computers in Japan 23 (2), 101–112.

Mavridou, T., Pardalos, P., Pitsoulis, L., Resende, M., 1998. A GRASP for the bi-quadratic assignment problem. European Journal of Operational Research 105, 613–621.

Melamed, I., Sergeev, S., Sigal, I., 1990. Traveling salesman problem: Theoretical issues. Automation Remote Control 50 (9), 1147–1173.

Modares, A., Somhom, S., Enkawa, T., 1999. A self-organizing neural network approach for multiple traveling salesman and vehicle routing problems. International Transactions in Operational Research 6, 591–606.

Moraga, R., AlMazid, M., 2000. The traveling salesman problem (TSP) and its industrial applications. Proceeding of the 4th International Conference on Engineering Design and Automation. p. 474–480.

Padberg, M., Rinaldi, G., 1991. A branch-and-cut algorithm for the solution of large-scale traveling salesman problems. SIAM Review 33, 60–100.

Rego, C., 1998. Relaxed tours and path ejections for the traveling salesman problem. European Journal of Operational Research 106 (2–3), 522–538.

Reinelt, G., 1994. The traveling salesman: computational solutions for TSP applications. Lectures Notes in Computer Science, Springer-Verlag.

Reinelt, G., Bixby, W., 1995. TSPLIB—a library of traveling salesman and related problem instances. Available from <http://www.crpc.rice.edu/softlib/catalog/tsplib.html>.

Renaud, J., Boctor, F.F., Laporte, G., 1996. A fast composite heuristic for the symmetric traveling salesman problem. INFORMS Journal on Computing 8 (2), 134–143.

Rios-Mercado, R., Bard, J., 1998. Heuristics for the flow line problem with setup costs. European Journal of Operational Research 1, 76–98.

Schmitt, L., Amini, M., 1998. Performance characteristics of alternative genetic algorithmic approaches to the Traveling Salesman Problem using path representation: An empirical study. European Journal of Operational Research 108 (3), 551–580.

Schulman, N., 2001. Meta-RaPS: A randomized priority search applied to the traveling salesman problem. M. Eng. IE. Thesis, Department of Industrial Engineering, University of Louisville, Louisville, Kentucky.

Sosnowska, D., 2000. Optimization of a simplified fleet assignment problem with metaheuristics: simulated annealing and GRASP. In: Pardalos, P. (Ed.), Approximation and Complexity in Numerical Optimization. Kluwer Academic Publishers.

Soylu, M., Ozdemirel, N., Kayaligil, S., 2000. A self-organizing neural network approach for the single AGV routing problem. European Journal of Operational Research 121 (1), 124–137.

Tsubakitani, S., Evans, J.R., 1998. An empirical study of a new metaheuristic for the traveling salesman problem. European Journal of Operational Research 104, 113–128.

Voudouris, C., Tsang, E., 1999. Guided local search and its application to the traveling salesman problem. European Journal of Operations Research 113, 469–499.

Yip, P.P.C., Pao, Y.H., 1995. Combinatorial optimization with use of guided evolutionary simulated annealing. IEEE Transactions on Neural Networks 6 (2), 290–295.