Gilbert Babin, Stéphanie Deneault, and Gilbert Laporte. *Improvements to the Or-opt Heuristic for the Symmetric Traveling Salesman Problem*. Cahier du GERAD no. G-2005-02. GERAD - Group for Research in Decision Analysis. Montréal, Québec, Canada. January 2005.

Improvements to the Or-opt Heuristic for the Symmetric Traveling Salesman Problem

Gilbert Babin[*]
Stéphanie Deneault[†]
Gilbert Laporte[†]

January, 2005

Abstract

Several variants and generalizations of the Or-opt heuristic for the *Symmetric Traveling Salesman Problem* are developed and compared on random and planar instances. Some of the proposed algorithms are shown to significantly improve upon the standard 2-opt and Or-opt heuristics.
Key words: symmetric traveling salesman problem, local search heuristics.

[*]Information Technology, HEC Montréal, 3000 chemin de la Côte-Sainte-Catherine, Montréal, Canada, H3T 2A7.

[†]Canada Research Chair in Distribution Management, HEC Montréal, 3000 chemin de la Côte-Sainte-Catherine, Montréal, Canada, H3T 2A7.

Gilbert Babin, Stéphanie Deneault, and Gilbert Laporte. *Improvements to the Or-opt Heuristic for the Symmetric Traveling Salesman Problem*. Cahier du GERAD no. G-2005-02. GERAD - Group for Research in Decision Analysis. Montréal, Québec, Canada. January 2005.

1 Introduction

The *Symmetric Traveling Salesman Problem* (STSP) is defined on a graph $G = (V, E)$, where $V = \{1, \dots, n\}$ is a vertex set and $E = \{(i, j) : i, j \in V, i < j\}$ is an edge set. With each edge (i, j) is associated a cost c_{ij}. The STSP consists of determining a least cost Hamiltonian cycle (or tour) on G. Over the past 50 years several exact and heuristic algorithms have been proposed for this problem. Branch-and-cut methods constitute the favorite exact solution methodology. These are rooted in the semilar paper of Dantzig, Fulkerson and Johnson (1954) and have culminated recently in sophisticated implementations capable of handling instances with thousands of vertices (Applegate *et al.*, 2003). For surveys see Lawler *et al.* (1985) and Gutin and Punnen (2002).

Several heuristics have also been proposed for the STSP. Here we focus on two main classes of tour improvement mechanisms: edge exchanges (EE) and chain exchanges (CE). The most famous EE method is called r-opt. At a given iteration it removes r edges from the current tour and attempts to find a better reconnection of the r remaining chains. The complexity of an r-opt iteration is $O(n^r)$. It is Croes (1958) who proposed the first systematic 2-opt method while the generalized concept was put forward by Lin (1965). A dynamic r-opt heuristic in which the value of r is allowed to vary during the search was developed by Lin and Kernighan (1973). Sophisticated implementations of this approach have been devised by Johnson and McGeoch (1997, 2002) and by Helsgaun (2000). These probably constitute the best available heuristics for the STSP. One of the best known CE methods, called Or-opt, is due to Or (1976). It attempts to improve the current tour by first moving a chain of three consecutive vertices in a different location (and possibly reversing it) until no further improvement can be obtained. The process is then repeated with chains of two consecutive vertices, and then with single vertices. Each iteration of Or-opt requires $O(n^2)$ operations.

Despite the computational superiority of some Lin-Kernighan implementations, simpler heuristics such as 2-opt, 3-opt and Or-opt remain popular due to their ease of implementation and their reasonably good performance. These are often used as a subroutine within improvement heuristics for more involved problems, such as the *Vehicle Routing Problem* (Laporte and Semet, 2002). A common step in a vehicle routing heuristic is to post-optimize individual vehicle routes. Because this operation is frequently applied, it makes sense to seek a fast but not necessarily highly accurate tour improvement heuristic.

The purpose of this paper is to develop and compare a number of Or-opt variants for the STSP. As will be shown, several versions are superior to the original implementation made by Or, and some 2-opt and Or-opt hybrids are also quite appealing. When conducting this study we have limited our options to "low complexity" heuristics, i.e., EE or CE heuristics for which the complexity of an iteration is $O(n^2)$. We have also restricted ourselves to the symmetric version of the problem because the heuristics under consideration are implemented quite differently in the asymmetric case and their computational behaviour is also

Gilbert Babin, Stéphanie Deneault, and Gilbert Laporte. *Improvements to the Or-opt Heuristic for the Symmetric Traveling Salesman Problem*. Cahier du GERAD no. G-2005-02. GERAD - Group for Research in Decision Analysis. Montréal, Québec, Canada. January 2005.

Table 1: FI/BI solution cost and CPU time ratios

	FI/BI Cost	FI/BI CPU
2-opt (random)	1.03***	0.26***
2-opt (planar)	0.99***	1.10***
Or-opt (random)	0.98***	1.21***
Or-opt (planar)	1.00	1.09***

*** indicates a 0.005 significance level.

different (Johnson *et al.*, 2002).

In Section 2 we describe some variants of the Or-opt heuristic. Computational results and analyses are presented in Section 3.

2 Variants of the Or-opt heuristic

There exist a number of misconceptions and contradictions regarding the Or-opt heuristic. Golden and Stewart (1985) who applied it as a post-optimizer after their CCA construction procedure conclude: “In general, the Or-opt procedure yields solutions that are comparable to the 3-opt in terms of quality of solution in an amount of time closer to that of the 2-opt procedure” (page 232). In contrast, Johnson and McGeoch (2002) who performed an extensive comparison of several STSP heuristics state that “Or-opt no longer appears to be a serious competitor” (page 414). As will be seen, our own tests indicate that 2-opt provides significantly better solutions than Or-opt when two basic and similarly implemented versions of these heuristics are applied to an arbitrarily generated initial solution.

The number of ways to implement even the most basic STSP heuristics is very large. In addition, computational comparisons may be tainted not only by implementation choices but also by the instances used to conduct the tests. One basic alternative when assessing the quality of an improvement heuristic is whether to start from an arbitrary tour or from a “good” tour produced by a construction method. Another question is whether to implement the first improving (FI) move at each iteration or the best improving (BI) move. These issues were partially answered by Hansen and Mladenović (2004) who concluded that for 2-opt, FI is slightly better but much faster than BI if one starts from an arbitrary tour, but the reverse result holds if a good tour is used as a starting point. We have implemented the FI and BI variants of 2-opt and Or-opt on 1000 random and planar 100-vertex instances starting from an arbitrary solution. Our results are summarized in Table 1. Following these tests we have opted to conduct all our experiments with FI which generally provides better costs with Or-opt.

In addition to these basic implementation alternatives, Johnson and McGeoch (2002) describe four speed up rules: 1) avoiding search space redundancies: when implementing r-opt, time can be saved by avoiding some exchanges that cannot improve the solution; 2) bounded neighbour lists: only consider the p closest neighbours of a vertex when performing reconnections (this rule was implemented by Zweig, 1995); 3) "don't look" bits: avoid considering certain moves if such moves have proved fruitless in the past; 4) tree-based tour representation: use a tree-based representation of the tour to accelerate computations.

A first contribution of this article is to generalize the Or-opt algorithm to allow for the relocation of chains of length exceeding 3. We denote by Or$(k_1, \ldots, k_s)$ an implementation of Or-opt which successively relocates chains of length $k_1, \ldots, k_s$. Thus the classical Or-opt algorithm is Or(3,2,1).

A second contribution applicable to Or$(k_1, \ldots, k_s)$ is to consider every k_t-length chain rooted at the same vertex i before moving on to the successor of i. We call this variant *vertex first* (VF) as opposed to the standard *tour first* (TF) implementation, and we denote it as Or $(k_1, \ldots, k_s)^{\mathrm{VF}}$. The TF and VF implementations of an Or-opt iteration using FI are described as follows.

Iteration of Or$(k_1, \ldots, k_s)$

```
for each $k_t$
   for each vertex $i$ in tour
      for each edge $e$ remaining in tour
         compute cost of removing $e$ and reconnecting tour
                        with $k_t$-length chain rooted at $i$
         if improvement found
            implement improvement
            break loop on $i$
         end if
      end for
   end for
end for
```

Iteration of Or$(k_1, \ldots, k_s)^{\mathbf{VF}}$

```
for each vertex $i$ in tour
   for each $k_t$
      for each edge $e$ remaining in tour
         compute cost of removing $e$ and reconnecting tour
                        with $k_t$-length chain rooted at $i$
         if improvement found
```

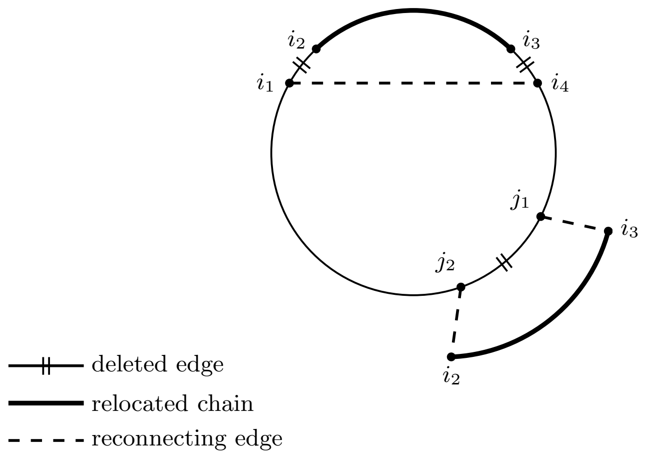

Figure 1: Cost impact of a chain relocation for a given orientation

```
                implement improvement
                break loop on $k_t$
            end if
        end for
    end for
end for
```

A third contribution is to avoid performing some chain relocations when the expected gain is deemed insufficient. As shown in Figure 1, removing the chain $(i_2, \ldots, i_3)$ and reconnecting i_1 with i_4 generates a gain equal to $g = c_{i_1 i_2} + c_{i_3 i_4} - c_{i_1 i_4}$. Similarly, relocating the chain between j_1 and j_2 (in the depicted orientation) yields a loss equal to $\ell = c_{i_2 j_2} + c_{i_3 j_1} - c_{j_1 j_2}$. While g is known as soon as the chain is considered for relocation, the incurred loss ℓ is not known *a priori* because it depends on the pair of vertices j_1 and j_2 between which the chain will be relocated. However, one can roughly approximate the average value of ℓ as $\bar{\ell} = 2\bar{c} - \bar{d}$, where $\bar{c}$ is the average edge cost and $\bar{d}$ is the average cost of an edge on the tour. In our implementation we perform a chain relocation only if $g > \lambda\bar{\ell}$, where λ is a user-controlled parameter. A larger value of λ means that fewer exchanges will be attempted, resulting in shorter computing times but possibly worse solutions. This Or-opt variant will be denoted $\mathrm{Or}(k_1, \ldots, k_s)^{\lambda}$ or $\mathrm{Or}(k_1, \ldots, k_s)^{\lambda,\mathrm{VF}}$, depending on whether TF or VF is applied. In the above pseudo-code descriptions, the central operation becomes:

```
if $g > \lambda\bar{\ell}$
    compute cost of removing $e$ and reconnecting tour
                        with $k_t$-length chain rooted at $i$
    if improvement found
```

Gilbert Babin, Stéphanie Deneault, and Gilbert Laporte. *Improvements to the Or-opt Heuristic for the Symmetric Traveling Salesman Problem*. Cahier du GERAD no. G-2005-02. GERAD - Group for Research in Decision Analysis. Montréal, Québec, Canada. January 2005.

```
        implement improvement
        break loop on $i$ (TF) or $k_t$ (VF)
      end if
    end if
```

This test is somewhat similar to the "don't look" bit just described except that it is applied to the likely net gain associated with a future move as opposed to a cost already observed in the past.

A fourth contribution of this article is to assess the empirical performance of the new Or-opt variants. These include $\text{Or}(k_1, \ldots, k_s)$, $\text{Or}(k_1, \ldots, k_s)^p$, $\text{Or}(k_1, \ldots, k_s)^\lambda$, $\text{Or}(k_1, \ldots, k_s)^{\text{VF}}$, and $\text{Or}(k_1, \ldots, k_s)^{\lambda,\text{VF}}$. The $\text{Or}(k_1, \ldots, k_s)^p$ variant denotes a bounded neighbour list heuristic, such as the one used by Zweig, p being the number of neighbours considered. We have also studied different hybrid heuristics $H_1 + H_2$ consisting of iteratively applying heuristic H_1 and heuristic H_2. Using this notation, the Bentley (1992) 2.5-opt heuristic is referred to as 2-opt+Or(1). Comparisons of the proposed algorithms were made with $\text{Or}(3, 2, 1)$, and the best algorithms were also compared with 2-opt.

3 Computational results

The procedures just described were coded in C and run on a Sun Ultra-10 computer (300 MHz). We have generated two classes of instances: random instances in which the c_{ij} values were generated according to a discrete uniform distribution on $[0, 100]$, and planar instances in which (X_i, Y_i) coordinates were first generated according to a discrete uniform distribution at integer coordinate points of the $[0, 100]^2$ square and each c_{ij} was computed as $\lfloor[(X_i - X_j)^2 + (Y_i - Y_j)^2]^{\frac{1}{2}}\rfloor$. In random instances the expected cost $\bar{c}$ is equal to 50 whereas in planar instances it is 52.23. The value of $\bar{d}$ is dynamically updated as the average edge cost on the current tour. In each case we have generated 1000 instances of size $n = 100$. All reported statistics are average values over the 1000 instances. We have performed left-tailed significance tests for paired comparisons (test 10 in Kanji, 1994). The significance levels are * ($\alpha = 0.025$), ** ($\alpha = 0.01$), and *** ($\alpha = 0.005$).

We have conducted several experiments to determine the most suitable value of λ. As expected, a larger λ value produces higher costs and shorter CPU times. Thus, setting $\lambda = 0.25$ yields slightly shorter times but much worse solutions; conversely, setting $\lambda = 0$ does not have much impact on solution quality but requires larger CPU times. We found that $\lambda = 0.1$ offers a good compromise for both random and planar instances. In the $\text{Or}(k_1, \ldots, k_s)^p$ experiments, we have used $p = 15$ as in Zweig (1995).

Tables 2 and 3 provide computational results for five $\text{Or}(k_1, \ldots, k_s)$ variants over random and planar instances, respectively. All entries are average values normalized with respect

Table 2: Comparison of several variants of the Or-opt heuristic on random instances (average statistics over 1000 instances normalized with respect to Or$(3, 2, 1)$)

	Or$(k_1, \dots, k_s)$		Or$(k_1, \dots, k_s)^{p=15}$		Or$(k_1, \dots, k_s)^{\lambda=0.1}$		Or$(k_1, \dots, k_s)^{\mathrm{VF}}$		Or$(k_1, \dots, k_s)^{\lambda=0.1,\mathrm{VF}}$	
$k_1, \dots, k_s$	Cost	CPU	Cost	CPU	Cost	CPU	Cost	CPU	Cost	CPU
1	173.74	**49.32*****	321.09	**47.71*****	264.83	**1.56*****	173.22	**51.56*****	267.89	**1.78*****
2	130.11	**59.33*****	500.00	**47.22*****	219.44	**1.23*****	129.53	**61.02*****	221.50	**1.95*****
3	119.93	**61.87*****	481.05	**47.23*****	193.63	**1.87*****	119.40	**61.58*****	192.67	**1.62*****
4	113.39	**64.00*****	481.13	**47.31*****	177.31	**2.43*****	112.75	**65.67*****	178.18	**2.67*****
2,1	119.25	**77.13*****	246.72	**48.34*****	186.55	**2.44*****	110.23	102.97	168.70	**2.74*****
3,2,1	*100.00*	*100.00*	213.32	**49.20*****	149.94	**2.79*****	**88.74*****	153.41	131.10	**4.16*****
4,...,1	**89.90**[1]***	118.14	196.30	**50.10*****	131.06	**3.40*****	**77.25*****	196.68	112.12	**4.78*****
5,...,1	**83.23*****	134.12	183.40	**50.76*****	119.56	**3.32*****	**70.26*****	248.84	100.28	**5.62*****
9,7,5,3,1	**78.39*****	131.29	185.44	**51.32*****	110.13	**4.23*****	**98.28*****	**69.17*****	145.50	**2.87*****
10,8,6,4,2	**76.99*****	132.91	206.32	**50.95*****	107.53	**4.45*****	**96.85*****	**68.95*****	141.59	**2.94*****
10,8,6,4,2,1	**75.65*****	141.75	173.56	**52.27*****	104.83	**5.10*****	**96.85*****	**68.88*****	141.46	**3.20*****
10,...,1	**67.86*****	190.26	146.50	**55.57*****	**93.57*****	**5.17*****	**55.69*****	447.80	**77.72*****	**11.22*****
15,...,1	**61.72*****	226.06	129.17	**61.58*****	**83.74*****	**7.14*****	**50.51*****	622.81	**69.77*****	**15.13*****
25,...,1	**55.79*****	283.56	110.54	**77.15*****	**75.21*****	**10.79*****	**46.64*****	892.14	**62.98*****	**26.52*****
50,...,1	**51.05*****	372.84	**92.49*****	130.09	**67.85*****	**20.33*****	**44.26*****	1224.69	**57.82*****	**54.50*****
1,2,3	106.63	114.48	218.24	**49.10*****	163.65	**3.23*****	**88.53*****	155.12	130.78	**4.82*****
1,...,10	**72.24*****	224.02	143.57	**55.28*****	101.23	**6.01*****	**55.55*****	453.55	**77.90*****	**12.46*****
1,...,15	**65.15*****	272.71	125.89	**61.27*****	**89.89*****	**6.98*****	**50.66*****	617.19	**69.75*****	**13.97*****
1,...,25	**58.20*****	346.60	108.40	**76.25*****	**79.01*****	**10.73*****	**46.48*****	899.90	**62.81*****	**26.18*****
1,...,50	**52.31*****	467.27	**91.42*****	128.11	**69.96*****	**21.14*****	**44.18*****	1249.44	**57.69*****	**52.83*****
1,2,3,2,1	**97.17*****	143.95	165.14	**50.68*****	145.67	**3.61*****	173.22	**51.27*****	268.10	**1.87*****
3,2,1,2,3	**93.09*****	126.12	183.19	**50.51*****	137.78	**3.31*****	119.40	**61.81*****	192.32	**2.29*****
Average	88.26	174.44	213.77	60.82	130.56	5.97	88.57	355.66	133.12	11.64

[1] Values in **bold** are better than Or$(3, 2, 1)$.

Table 3: Comparison of several variants of the Or-opt heuristic on planar instances (average statistics over 1000 instances normalized with respect to Or$(3,2,1)$)

	Or$(k_1,\ldots,k_s)$		Or$(k_1,\ldots,k_s)^{p=15}$		Or$(k_1,\ldots,k_s)^{\lambda=0.1}$		Or$(k_1,\ldots,k_s)^{\mathrm{VF}}$		Or$(k_1,\ldots,k_s)^{\lambda=0.1,\mathrm{VF}}$	
$k_1,\ldots,k_s$	Cost	CPU	Cost	CPU	Cost	CPU	Cost	CPU	Cost	CPU
1	116.48	**81.71*****	272.07	**43.77*****	166.25	**4.89*****	116.04	**90.69*****	167.04	**4.67*****
2	110.08	**65.86*****	391.46	**43.33*****	140.14	**4.68*****	109.98	**68.50*****	138.64	**4.96*****
3	112.85	**60.32*****	425.68	**43.32*****	134.01	**4.80*****	112.40	**64.38*****	132.99	**4.88*****
4	116.37	**57.65*****	429.63	**43.43*****	133.92	**4.46*****	116.20	**61.32*****	133.30	**4.69*****
2,1	104.17	**86.14*****	231.64	**44.66*****	127.88	**6.36*****	104.27	116.21	124.50	**7.13*****
3,2,1	*100.00*	*100.00*	196.12	**45.66*****	114.59	**6.95*****	100.99	142.48	113.04	**8.86*****
4,. . . ,1	**97.99**[1]***	117.10	172.80	**46.48*****	108.57	**8.50*****	**98.49*****	174.71	107.65	**10.50*****
5,. . . ,1	**97.08*****	134.78	157.45	**47.24*****	105.62	**9.38*****	**97.32*****	203.67	104.84	**11.29*****
9,7,5,3,1	**96.65*****	146.16	166.83	**47.16*****	104.91	**10.60*****	130.64	**62.16*****	143.07	**4.76*****
10,8,6,4,2	100.88	149.11	176.71	**46.89*****	108.95	**10.48*****	132.78	**61.69*****	144.42	**5.04*****
10,8,6,4,2,1	**96.09*****	162.93	150.97	**48.11*****	103.28	**11.20*****	132.78	**61.67*****	144.50	**5.18*****
10,. . . ,1	**95.36*****	214.14	119.93	**51.41*****	101.30	**14.28*****	**94.73*****	354.94	**99.53*****	**17.27*****
15,. . . ,1	**94.95*****	270.93	108.95	**56.53*****	100.33	**16.94*****	**93.87*****	509.32	**97.74*****	**23.25*****
25,. . . ,1	**94.73*****	340.55	103.22	**68.39*****	**99.67***	**21.79*****	**93.34*****	786.93	**96.64*****	**36.69*****
50,. . . ,1	**94.23*****	406.87	100.41	108.77	**98.88*****	**25.96*****	**93.01*****	1306.97	**95.96*****	**71.85*****
1,2,3	107.43	123.29	174.53	**45.43*****	128.04	**7.49*****	101.62	184.97	114.22	**9.57*****
1,. . . ,10	103.75	177.16	130.45	**50.81*****	115.24	**10.88*****	**95.72*****	390.62	100.51	**16.13*****
1,. . . ,15	102.72	205.59	124.82	**55.07*****	112.91	**11.62*****	**94.59*****	518.42	**98.57*****	**19.95*****
1,. . . ,25	101.43	256.72	119.74	**65.48*****	110.14	**14.07*****	**93.70*****	744.85	**97.11*****	**30.71*****
1,. . . ,50	**99.82**	352.49	115.12	102.16	107.18	**20.76*****	**93.14*****	1156.73	**96.27*****	**58.19*****
1,2,3,2,1	104.35	147.96	138.79	**47.22*****	119.40	**9.08*****	116.04	**92.90*****	167.07	**4.69*****
3,2,1,2,3	**99.40*****	115.75	151.56	**46.95*****	112.45	**7.72*****	112.40	**65.76*****	133.08	**4.76*****
Average	102.13	171.51	189.04	54.47	116.08	11.04	106.09	328.18	120.49	16.59

[1] Values in **bold** are better than Or$(3,2,1)$.

Gilbert Babin, Stéphanie Deneault, and Gilbert Laporte. *Improvements to the Or-opt Heuristic for the Symmetric Traveling Salesman Problem*. Cahier du GERAD no. G-2005-02. GERAD - Group for Research in Decision Analysis. Montréal, Québec, Canada. January 2005.

Gilbert Babin, Stéphanie Deneault, and Gilbert Laporte. *Improvements to the Or-opt Heuristic for the Symmetric Traveling Salesman Problem*. Cahier du GERAD no. G-2005-02. GERAD - Group for Research in Decision Analysis. Montréal, Québec, Canada. January 2005.

to $\text{Or}(3,2,1)$. We first comment on random instances. The $\text{Or}(k_1,\ldots,k_s)$ column of Table 2 shows that all variants with $s \geq 4$ are significantly better than $\text{Or}(3,2,1)$ in terms of cost at the expense of an increased CPU time. This is a direct consequence of the fact that the number of solutions explored during the search increases with s. The best variants are those larger values of s with strictly decreasing or increasing k_t values. The $\text{Or}(k_1,\ldots,k_s)^{p=15}$ variant reduces CPU times by a factor of about 3 but about doubles the solution costs when compared with $\text{Or}(k_1,\ldots,k_s)$. As expected, $\text{Or}(k_1,\ldots,k_s)^{\lambda=0.1}$ yields larger solution values than $\text{Or}(k_1,\ldots,k_s)$ but smaller CPU times by at least one order of magnitude. It also appears to dominate $\text{Or}(k_1,\ldots,k_s)^{p=15}$ in terms of solution quality and computing time. The $\text{Or}(k_1,\ldots,k_s)^{\text{VF}}$ variant seems to improve the $\text{Or}(k_1,\ldots,k_s)$ solution values when the k_t values are increasing or decreasing, and consecutive. However, this improvement comes at the price of roughly doubling the average CPU time. Finally, the $\text{Or}(k_1,\ldots,k_s)^{\lambda=0.1,\text{VF}}$ combination improves the $\text{Or}(k_1,\ldots,k_s)^{\lambda=0.1}$ cost results when $\text{Or}(k_1,\ldots,k_s)^{\lambda=0.1}$ was already better than $\text{Or}(3,2,1)$. When cost and CPU time are taken into account, $\text{Or}(k_1,\ldots,k_s)^{\lambda=0.1,\text{VF}}$ offers a good compromise between $\text{Or}(k_1,\ldots,k_s)^{\lambda=0.1}$ and $\text{Or}(k_1,\ldots,k_s)^{\text{VF}}$. Several of these observations extend to the planar case but then the cost reductions are typically less dramatic. The major difference between the random and planar results lies in the fact that $\text{Or}(k_1,\ldots,k_s)$ with $s \geq 4$, increasing and consecutive k_t values, behaves rather poorly in the planar case.

Tables 4 and 5 contain results on all heuristics of Tables 2 and 3 that dominate 2-opt and $\text{Or}(3,2,1)$, as well as on a number of dominating hybrid variants. These hybrids all execute 2-opt after a variant of Or-opt. We also tested hybrids using the reverse order, including the 2-opt+Or(1) hybrid of Bentley (1992), but none of these combinations dominated both 2-opt and $\text{Or}(3,2,1)$. Again, the best hybrids tend to contain long increasing or decreasing k_t sequences. They all use $\lambda = 0.1$ and often VF. The heuristics of Tables 4 and 5, and the efficient frontier are plotted in Figures 2 and 3. In the random case, the non-dominated algorithms are $\text{Or}(25,\ldots,1)^{\lambda=0.1}$, $\text{Or}(1,\ldots,15)^{\lambda=0.1,\text{VF}}$, and $\text{Or}(15,\ldots,1)^{\lambda=0.1,\text{VF}}$+2-opt. In the planar case, the non-dominated algorithms are $\text{Or}(15,\ldots,1)^{\lambda=0.1,\text{VF}}$+2-opt, $\text{Or}(25,\ldots,1)^{\lambda=0.1,\text{VF}}$+2-opt, $\text{Or}(50,\ldots,1)^{\lambda=0.1,\text{VF}}$+2-opt, and $\text{Or}(1,\ldots,50)^{\lambda=0.1,\text{VF}}$+2-opt. Only one algorithm, $\text{Or}(15,\ldots,1)^{\lambda=0.1,\text{VF}}$+2-opt, is non-dominated both for the random and planar cases. It is therefore recommended.

In closing, we note from Table 4 that very large improvements with respect to 2-opt and $\text{Or}(3,2,1)$ are obtained on random instances. This clearly indicates that these two heuristics are highly suboptimal on this class of instances. In particular, one should avoid applying them alone to combinatorial optimization problems with unstructured costs such as production scheduling problems with changeover penalties.

Gilbert Babin, Stéphanie Deneault, and Gilbert Laporte. *Improvements to the Or-opt Heuristic for the Symmetric Traveling Salesman Problem*. Cahier du GERAD no. G-2005-02. GERAD - Group for Research in Decision Analysis. Montréal, Québec, Canada. January 2005.

Table 4: Comparison of several variants of the Or-opt heuristic on random instances (average statistics over 1000 instances normalized with respect to 2-opt and $Or(3, 2, 1)$)

	vs 2-opt		vs $Or(3,2,1)$	
Algorithm[1]	Cost	CPU	Cost	CPU
2-opt	*100.00*	*100.00*	76.97***	27.68***
$Or(3,2,1)$	129.91	361.33	*100.00*	*100.00*
(1)[2] $\mathbf{Or(25,\ldots,1)^{\lambda=0.1}}$	97.70***	38.99***	75.21***	10.79***
(2) $Or(50,\ldots,1)^{\lambda=0.1}$	88.15***	73.46***	67.85***	20.33***
(3) $Or(1,\ldots,50)^{\lambda=0.1}$	90.88***	76.40***	69.96***	21.14***
(4) $Or(15,\ldots,1)^{\lambda=0.1,VF}$	90.65***	54.68***	69.77***	15.13***
(5) $Or(25,\ldots,1)^{\lambda=0.1,VF}$	81.82***	95.81	62.98***	26.52***
(6) $\mathbf{Or(1,\ldots,15)^{\lambda=0.1,VF}}$	90.61***	50.47***	69.75***	13.97***
(7) $Or(1,\ldots,25)^{\lambda=0.1,VF}$	81.60***	94.58	62.81***	26.18***
(8) $Or(10,\ldots,1)^{\lambda=0.1}$+2-opt	92.13***	67.24***	70.91***	18.61***
(9) $Or(15,\ldots,1)^{\lambda=0.1}$+2-opt	88.82***	68.20***	68.37***	18.88***
(10) $Or(25,\ldots,1)^{\lambda=0.1}$+2-opt	84.92***	79.18***	65.37***	21.91***
(11) $Or(1,\ldots,15)^{\lambda=0.1}$+2-opt	91.13***	87.14	70.15***	24.12***
(12) $Or(10,\ldots,1)^{\lambda=0.1,VF}$+2-opt	85.94***	66.88***	66.15***	18.51***
(13) $\mathbf{Or(15,\ldots,1)^{\lambda=0.1,VF}}$**+2-opt**	81.42***	72.70***	62.68***	20.12***
(14) $Or(1,\ldots,10)^{\lambda=0.1,VF}$+2-opt	86.40***	75.40***	66.50***	20.87***
(15) $Or(1,\ldots,15)^{\lambda=0.1,VF}$+2-opt	81.43***	83.70*	62.68***	23.16***

[1] Algorithms in **bold** are non-dominated.
[2] Numbers in parentheses refer to algorithms in Figures 2 and 3.

Table 5: Comparison of several variants of the Or-opt heuristic on planar instances (average statistics over 1000 instances normalized with respect to 2-opt and $Or(3, 2, 1)$)

	vs 2-opt		vs $Or(3,2,1)$	
Algorithm[1]	Cost	CPU	Cost	CPU
2-opt	*100.00*	*100.00*	96.83***	111.70
$Or(3,2,1)$	103.27	89.53***	*100.00*	*100.00*
(5)[2] $Or(25,\ldots,1)^{\lambda=0.1,VF}$	99.81	32.85***	96.64***	36.69***
(16) $Or(50,\ldots,1)^{\lambda=0.1,VF}$	99.10***	64.33***	95.96***	71.85***
(17) $Or(1,\ldots,50)^{\lambda=0.1,VF}$	99.42***	52.10***	96.27***	58.19***
(10) $Or(25,\ldots,1)^{\lambda=0.1}$+2-opt	97.76***	36.99***	94.66***	41.32***
(18) $Or(50,\ldots,1)^{\lambda=0.1}$+2-opt	97.53***	45.27***	94.44***	50.56***
(12) $Or(10,\ldots,1)^{\lambda=0.1,VF}$+2-opt	97.40***	29.33***	94.32***	32.77***
(13) $\mathbf{Or(15,\ldots,1)^{\lambda=0.1,VF}}$**+2-opt**	97.03***	28.64***	93.96***	31.99***
(19) $\mathbf{Or(25,\ldots,1)^{\lambda=0.1,VF}}$**+2-opt**	96.70***	45.98***	93.63***	51.36***
(20) $\mathbf{Or(50,\ldots,1)^{\lambda=0.1,VF}}$**+2-opt**	96.42***	74.34***	93.37***	83.03***
(15) $Or(1,\ldots,15)^{\lambda=0.1,VF}$+2-opt	97.15***	30.82***	94.07***	34.43***
(21) $Or(1,\ldots,25)^{\lambda=0.1,VF}$+2-opt	96.88***	37.07***	93.81***	41.40***
(22) $\mathbf{Or(1,\ldots,50)^{\lambda=0.1,VF}}$**+2-opt**	96.47***	64.93***	93.42***	72.52***

[1] Algorithms in **bold** are non-dominated.
[2] Numbers in parentheses refer to algorithms in Figures 2 and 3.

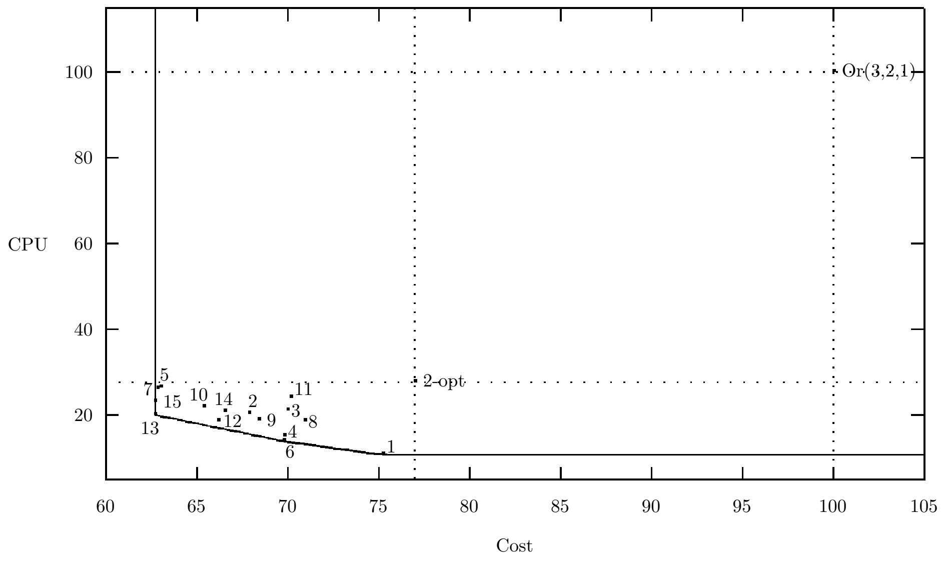

Figure 2: Costs and CPU times normalized with respect to Or(3, 2, 1) for the heuristics of Table 4 (random instances)

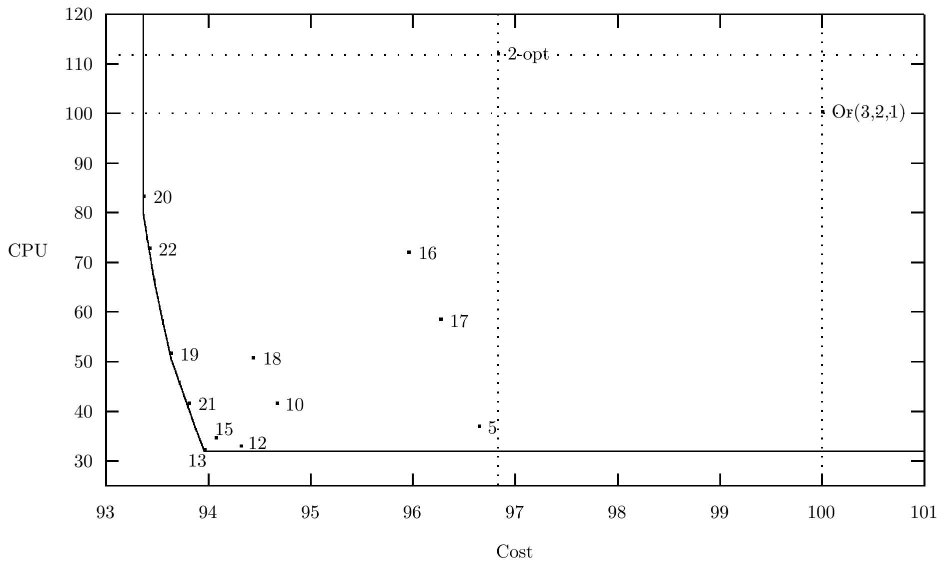

Figure 3: Costs and CPU times normalized with respect to Or(3, 2, 1) for the heuristics of Table 5 (planar instances)

Acknowledgements

This research was partially supported by the Canadian Natural Sciences and Engineering Research Council under grants 155899-03 and OGP0039682. This support is gratefully acknowledged.

References

Applegate, D., Bixby, R., Chvátal, V., Cook, W., Implementing the Dantzig-Fulkerson-Johnson algorithm for large scale traveling salesman problems, *Mathematical Programming Series B*, 2003; 97:91–153.

Bentley, J.L., Fast algorithms for geometric traveling salesman problems, *ORSA Journal on Computing*, 1992; 4:387–411.

Croes, G.A., A method for solving large scale symmetric traveling salesman problems to optimality, *Operations Research*, 1958; 6:791–812.

Dantzig, G.B., Fulkerson, D.R., Johnson, S.M., Solution of a large scale traveling salesman problem, *Operations Research*, 1954; 2:393–410.

Golden, B.L., Stewart, W.R., Empirical analysis of heuristics. In: Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., Shmoys, D.B., editors, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, Chichester: Wiley; 1985, p. 207–249.

Gutin, G., Punnen, A.P., editors, *The Traveling Salesman Problem and Its Variations*, Boston: Kluwer; 2002.

Hansen, P., Mladenović, N., First vs best improvement: An empirical study, *Discrete Applied Mathematics*, 2004; forthcoming.

Helsgaun, K., An effective implementation of the Lin-Kernighan traveling salesman heuristic, *European Journal of Operational Research*, 2000; 126:106–130.

Johnson, D.S., Gutin, G., McGeoch, L.A., Yeo, A., Zhang, W., Sversvitch, A., Experimental analysis of heuristics for the ATSP. In: Gutin, G., Punnen, A.P., editors, *The Traveling Salesman Problem and Its Variations*, Boston: Kluwer; 2002, p. 445–487.

Johnson, D.S., McGeoch, L.A., The traveling salesman problem: A case study. In: Aarts, E., Lenstra, J.K., editors, *Local Search in Combinatorial Optimization*, Chichester: Wiley; 1997, p. 215–310.

Johnson, D.S., McGeoch, L.A., Experimental analysis of heuristics for the STSP. In: Gutin, G., Punnen, A.P., editors, *The Traveling Salesman Problem and Its Variations*, Boston: Kluwer; 2002, p. 369–443.

Gilbert Babin, Stéphanie Deneault, and Gilbert Laporte. *Improvements to the Or-opt Heuristic for the Symmetric Traveling Salesman Problem*. Cahier du GERAD no. G-2005-02. GERAD - Group for Research in Decision Analysis. Montréal, Québec, Canada. January 2005.

Kanji, G.K., *100 Statistical Tests*, London: Sage; 1994.

Laporte, G., Semet, F.J., Classical heuristics for the capacitated VRP. In: Toth, P., Vigo, D., editors, *The Vehicle Routing Problem*, Philadelphia: SIAM Monographs on Discrete Mathematics and Applications; 2002, p. 109–128.

Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., Shmoys, D.B., editors, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, Chichester: Wiley; 1985.

Lin, S., Computer solutions of the traveling salesman problem, *Bell System Technical Journal*, 1965; 44:2245–2269.

Lin, S., Kernighan, B.W., An effective heuristic algorithm for the traveling salesman problem, *Operations Research*, 1973; 21:972–989.

Or, I., *Traveling Salesman-Type Combinatorial Problems and Their Relation to the Logistics of Regional Blood Banking.* Ph.D. Thesis, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL, 1976.

Zweig, G., An effective tour construction and improvement procedure for the traveling salesman problem, *Operations Research*, 1995; 43:1049–1047.