Parallel Local Search and the Travelling Salesman Problem

(Extended Abstract)

M.G.A. Verhoeven[a] E.H.L. Aarts[a,b] E. van de Sluis[a] R.J.M. Vaessens[a]

[a]Eindhoven University of Technology, Department of Mathematics and Computing Science, P.O. Box 513, 5600 MB Eindhoven, Netherlands

[b]Philips Research Laboratories, P.O. Box 80000, 5600 JA Eindhoven, Netherlands

Abstract

We present scalable parallel local search algorithms using multiple trial parallelism. The concept of distributed neighbourhood structures is introduced, and applied to the Travelling Salesman Problem (TSP). The performance of parallel local search algorithms for TSP instances with several thousands of cities is compared to the performance of the classical 2-opt, 3-opt, and Lin-Kernighan algorithms.

1. Introduction

In a combinatorial optimization problem one must find an optimal solution among a finite, possibly large, number of alternatives [1]. We consider optimization problems as minimization problems. An important achievement in combinatorial optimization is the existance of two classes of problems, $\mathcal{P}$ and $\mathcal{NP}$, and the fact that, unless $\mathcal{P} = \mathcal{NP}$, there exists a class of combinatorial optimization problems for which no algorithms exist that solve each instance of the problem to optimality in a time that is polynomially bounded in the size of the instance. These problems are called *NP-hard* [2]. As a consequence of this, the larger instances of a problem can only be solved to proximity in reasonable time.

Local search algorithms constitute a class of approximation algorithms that are based on repeatedly improving solutions. For many problems these algorithms find good-quality solutions. However, they often require substantial amounts of run-time for large instances. One way to reduce run-time is by using parallelism.

The Travelling Salesman Problem (TSP) is probably the best-known combinatorial optimization problem. Much of the theory for combinatorial optimization has been developed using the TSP as a proving ground [3]. The TSP is NP-hard, and considerable effort has been spent to design efficient approximation algorithms for it. The best results have been obtained by local search algorithms with iterated Lin-Kernighan as the winner [4]. Our goal is to design parallel local search algorithms that can compete with the best existing sequential algorithms.

2. Local search

A local search algorithm repeatedly tries to improve a given solution by searching its neighbourhood for a solution with lower cost.

Definition 2.1 *An instance of a combinatorial optimization problem is denoted as a pair $(\mathcal{S}, f)$, where $\mathcal{S}$ is the set of feasible solutions*—solution space—*and $f : \mathcal{S} \rightarrow \mathbb{R}$ is a function that gives the cost of a feasible solution. A* neighbourhood structure *$\mathcal{N} : \mathcal{S} \rightarrow \mathcal{P}(\mathcal{S})$ assigns to each feasible solution a set of solutions that can be directly obtained from it.*

Definition 2.2 *Let $t \in \mathcal{S}$, then t is a* local optimum *w.r.t. $\mathcal{N}$, if and only if*

$\forall_{t' \in \mathcal{N}(t)}\, f(t') \geq f(t)$.

A general local search algorithm can be formulated as follows.

proc *Local_Search*$(\mathcal{N} : \mathcal{S} \rightarrow \mathcal{P}(\mathcal{S}); t_0, t \in \mathcal{S}) =$
$[\![t, W := t_0, \emptyset$;
do $\mathcal{N}(t) \setminus W \neq \emptyset \longrightarrow$
$t' :\in \mathcal{N}(t) \setminus$ W ;
if $f(t') \geq f(t) \longrightarrow W := W \cup \{t'\}$
$[\!]\ f(t') < f(t) \longrightarrow t, W := t', \emptyset$
fi
od$\{$ t is a local optimum w.r.t. $\mathcal{N}$ $\}$
$]\!]$

2.1. Local search for the TSP

In the TSP, a salesman has to visit a prescribed list of cities exactly once and such that the length of the tour is minimal. The TSP can be defined as follows.

Definition 2.3 (TSP) *Let V be a set of N cities and $d(i,j) \in \mathbb{R}$ a distance for each $i, j \in V$. Let $\pi : \{0, \ldots, N-1\} \rightarrow V$ be a permutation of V. A* tour *$t \subseteq V \times V$ is a set of N edges that constitutes a Hamiltonian cycle in the complete graph $(V, V \times V)$ and is defined as $t = \{(\pi(s), \pi((s+1)\mathbf{mod} N)) \mid 0 \leq s < N\}$. A TSP instance is given by $(\mathcal{S}, f)$, where the solution space $\mathcal{S}$ is the set of all tours, and the cost function f is given by $f(t) = \sum_{(i,j) \in t} d(i,j)$. Find a tour $t \in \mathcal{S}$ for which $f(t)$ is minimal.*

Here we concentrate on the symmetric TSP, for which the distances satisfy $d(i,j) = d(j,i)$ for each $i, j \in V$. The best-known neighbourhood structures for the TSP are based on the exchange of a number of edges. We mention the following ones.

- *2-opt* $(\mathcal{N}_2)$. A tour t' is a neighbour of tour t, if t' can be obtained from t by removing two edges and inserting two edges. This transition is called a 2-exchange [5].

- *3-opt* $(\mathcal{N}_3)$. A tour t' is a neighbour of tour t, if t' can be obtained from t by removing three edges and inserting three edges. This transition is called a 3-exchange [5].

- *Lin-Kernighan* ($\mathcal{N}_{lk}$). Instead of exchanging a constant number of edges, this number varies and depends on the current tour t. Basically a series of 2-exchanges is constructed by a combination of first improvement, best improvement, and limited backtracking [6].

- *Or-opt* ($\mathcal{N}_{or}$). A tour t' is a neighbour of t, if it can be obtained from t by inserting a string of at most three adjacent cities between two other adjacent cities. This is a restrictive 3-exchange for which the length of the shortest sub-path is at most two [7].

The empirical average-case complexity of local search algorithms using the above neighbourhood structures is low order polynomial. Furthermore, good quality solutions can be found. They can be ranked, in order of decreasing average cost of final solutions, as follows: 2-opt, Or-opt, 3-opt, and Lin-Kernighan.

3. Parallel local search

In the design of parallel local search algorithms one may distinguish between *single trial parallelism* and *multiple trial parallelism* [8]. Here we concentrate on multiple trial parallelism. The idea is to generate and accept different transitions simultaneously. If a feasible solution can be partitioned into a number of disjoint partial solutions, the partial solutions can be improved simultaneously. Since transitions involving elements of different partial solutions are not generated in this way, the partitions have to be changed in the course of the search. In order to formalize this approach, we need a number of definitions.

Definition 3.1 *Assume that $t \in \mathcal{S}$ is a set. Define a* partial solution *τ as $\tau \subseteq t$, and the set of all partial solutions $\mathcal{U} = \bigcup_{t \in \mathcal{S}} \mathcal{P}(t)$.*

For the approach we have in mind, the cost difference between two neighbouring partial solutions must be computable independent from other partial solutions. Therefore, an instance $(\mathcal{S}, f)$ of a combinatorial optimization problem has to satisfy the following condition.

Condition 3.2 *A solution $t \in \mathcal{S}$ is a set, f can be extended to $f : \mathcal{U} \to I\!R$, and*

$$\forall_{t,t' \in \mathcal{S}} \forall_{\tau,\tau' \in \mathcal{U} \mid \tau \subseteq t \,\wedge\, \tau' \subseteq t' \,\wedge\, t \setminus \tau = t' \setminus \tau'} \; f(t) - f(t') = f(\tau) - f(\tau').$$

If $(\mathcal{S}, f)$ satisfies Condition 3.2, we introduce the following definitions to design a parallel local search algorithm for it.

Definition 3.3 *Let $t \in \mathcal{S}$ and $P \geq 1$. Define the set of processes $\Omega = \{0, \ldots, P-1\}$, a* distribution *$\delta : \Omega \to \mathcal{U}$, a* local neighbourhood structure *$\mathcal{L} : \mathcal{U} \to \mathcal{P}(\mathcal{U})$ with $\tau \in \mathcal{L}(\tau)$ for all $\tau \in \mathcal{U}$, and a* distribution structure *Δ, where $\Delta(t, P) = \{\delta : \Omega \to \mathcal{U} \mid \{\delta(p) \mid p \in \Omega\}$ is a partition of $t\}$. Then, a* distributed neighbourhood structure *$\mathcal{N}_P$ is a pair $(\Delta, \mathcal{LN})$, where Δ is a distribution structure and $\mathcal{LN} : \Omega \to (\mathcal{U} \to \mathcal{P}(\mathcal{U}))$.*

According to Definition 3.3 a partial solution is a subset of a feasible solution, and a distribution is a mapping that assigns to each process a partial solution. A local neighbourhood structure assigns to each partial solution a set of partial solutions—a *local neighbourhood*—that can be obtained from it. A distribution structure assigns to each $t \in \mathcal{S}, P \geq 1$ a set of distributions where the range of each distribution is a partition of t in P partial solutions. A distributed neighbourhood structure consists of a distribution structure and a mapping that assigns a local neighbourhood structure to each process. Next, we formalize the notion of local optimality for a distributed neighbourhood structure.

Definition 3.4 *Let $\mathcal{L}$ be a local neighbourhood structure and τ a partial solution, then τ is a* local local optimum *w.r.t. $\mathcal{L}$, if and only if*

$\forall_{\tau' \in \mathcal{L}(\tau)}\, f(\tau') \geq f(\tau).$

Definition 3.5 *Let $\mathcal{N}_P$ be a distributed neighbourhood structure and $t \in \mathcal{S}$, then t is a* global local optimum *w.r.t. $\mathcal{N}_P$, if and only if*

$\forall_{\delta \in \Delta(t,P)} \forall_{p \in \Omega}\, \delta(p)$ *is a local local optimum w.r.t. $\mathcal{LN}(p)$.*

Using these definitions we can state the following general parallel local search algorithm.

proc *Parallel_Local_Search*$(t \in \mathcal{S}) =$
$\quad[\![D := \emptyset$;
$\quad$ **do** $\Delta(t,P) \setminus D \neq \emptyset \rightarrow$
$\quad\quad \delta :\in \Delta(t,P) \setminus D$;
$\quad\quad$ $\mathbf{par}_{p \in \Omega}$ *Local_Search*$(\mathcal{LN}(p), \delta(p), \delta'(p))$ **rap**;
$\quad\quad$ **if** $\forall_{p \in \Omega}\, f(\delta'(p)) = f(\delta(p)) \rightarrow D := D \cup \{\delta\}$
$\quad\quad [\!]\ \exists_{p \in \Omega}\, f(\delta'(p)) < f(\delta(p)) \rightarrow D, t := \emptyset, \bigcup_{p \in \Omega} \delta'(p)$
$\quad\quad$ **fi** ;
$\quad$ **od**$\{$ t is a global local optimum w.r.t. $\mathcal{N}_P$ $\}$
$\quad]\!]$

The algorithm terminates, because the number of solutions is finite. We still have to prove that a global local optimum is feasible i.e. belongs to the solution space. Therefore, we require that each solution found by the algorithm, is feasible. This is more strict than necessary, for we only need to prove the feasibility of the final solution. The feasibility of each solution found by the algorithm is guaranteed by the *safety* of a distributed neighbourhood structure. The safety of a distributed neighbourhood structure expresses that the composition of local neighbours of partial solutions is a feasible solution.

Definition 3.6 *A distributed neighbourhood structure $(\Delta, \mathcal{LN})$ is* safe*, if and only if*

$\forall_{\delta \in \Delta(t,P)} \forall_{\delta' : \Omega \rightarrow \mathcal{U} | \forall_{p \in \Omega} \delta'(p) \in \mathcal{LN}(p)(\delta(p))} \left(\bigcup_{q \in \Omega} \delta'(q) \in \mathcal{S}\right).$

An important issue is raised by the question how we can compare a distributed neighbourhood structure $\mathcal{N}_P$ with a neighbourhood structure $\mathcal{N}$. This enables us to compare the solutions found by a parallel local search algorithm with the solutions found by a sequential local search algorithm. For this purpose we introduce the following definitions.

Let $\mathcal{N}$ be a neighbourhood structure for an instance $(\mathcal{S}, f)$ of a combinatorial optimization problem and $\mathcal{N}_P$ a distributed neighbourhood structure for this instance, then

Definition 3.7 *$\mathcal{N}_P$ is called* epimorphic *to $\mathcal{N}$, if and only if*

$\forall_{t\in\mathcal{S}}\, t$ is a global local optimum w.r.t. $\mathcal{N}_P \Rightarrow t$ is a local optimum w.r.t. $\mathcal{N}$.

Definition 3.8 *$\mathcal{N}_P$ is called* isomorphic *to $\mathcal{N}$, if and only if*

$\forall_{t\in\mathcal{S}}\, t$ is a global local optimum w.r.t. $\mathcal{N}_P \Leftrightarrow t$ is a local optimum w.r.t. $\mathcal{N}$.

These definitions are used to compare a sequential and a parallel local search algorithm with respect to the quality of the final solutions. The following observations hold.

- If $\mathcal{N}_P$ is epimorphic to $\mathcal{N}$, then the expected cost of the solutions found by a sequential local search algorithm is an upper bound for the expected cost of the solutions found by a parallel local search algorithm.

- If $\mathcal{N}_P$ is isomorphic to $\mathcal{N}$, then the expected cost of the solutions found by a sequential local search algorithm is equal to the expected cost of the solutions found by a parallel local search algorithm.

4. Distributed neighbourhood structures for the TSP

The notions that we have introduced, have been applied to the TSP. Here we restrict ourselves to presenting the main results; a technical treatment can be found in [9]. In the TSP, we define a *linear distribution structure* that gives a set of distributions in which each partial solution consists of two paths. Figure 1 gives an example of this distribution with four partial solutions. Different distributions are generated by "shifting" one edge to an adjacent partial solution.

Theorem 4.1 *A distributed neighbourhood structure with a linear distribution structure is safe, if each local neighbour, with addition of two edges, is a cycle.*

In [9] we define a distributed neighbourhood structure $\mathcal{N}_{2,P}$ that is isomorphic to $\mathcal{N}_2$, and therefore a parallel local search algorithm that uses $\mathcal{N}_{2,P}$ finds solutions with the same average cost as the solutions found by the sequential 2-opt algorithm.

Extensive experiments [4] have shown that 3-opt local search algorithms find solutions with considerable lower cost than solutions found by 2-opt local search algorithms.

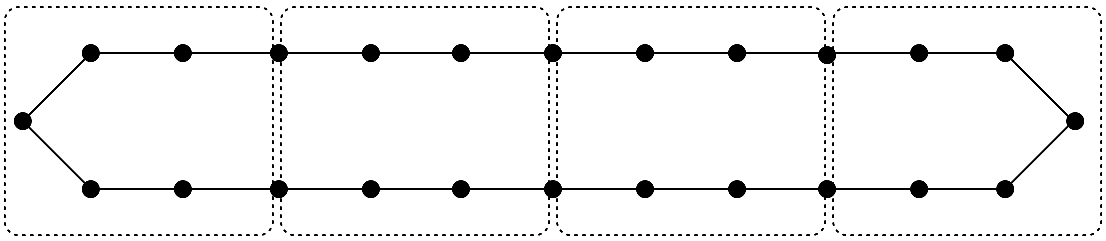

Figure 1. Linear distribution

Theorem 4.2 *No safe distributed neighbourhood structure $\mathcal{N}_P$ can exist, with partial solutions of identical size, that is epimorphic or isomorphic to $\mathcal{N}_3$.*

The size of partial solutions should be roughly equal to obtain an algorithm with a good load-balance. So, Theorem 4.2 shows that it is not possible to design a parallel local search algorithm with a linear speed-up that finds solutions with the same average cost as a sequential 3-opt algorithm. Therefore, we define $\mathcal{N}_{3,B}$ as a restricted version of $\mathcal{N}_3$. The neighbours of a solution in $\mathcal{N}_{3,B}$ are tours that can be obtained by a 3-exchange for which the length of the shortest sub-path is at most B. It can be proven that $\mathcal{N}_{or}(t) \subseteq \mathcal{N}_{3,4}(t)$ for $t \in \mathcal{S}$. In [9] we define distributed neighbourhood structures $\mathcal{N}_{3,B,P}$ and $\mathcal{N}'_{3,B,P}$ that are isomorphic and epimorphic to $\mathcal{N}_{3,B}$, respectively. We observe that verifying global local optimality of $\mathcal{N}'_{3,B,P}$ requires examination of $\mathcal{O}(\frac{N^3}{P} + B * N^2)$ tours, totally. Hence, the total number of tours examined to verify global local optimality, decreases with increasing P. As a consequence, the average cost of solutions found by a parallel algorithm that uses $\mathcal{N}'_{3,B,P}$ may increase with increasing P. The size of this effect is revealed by an empirical study (see Section 4).

The Lin-Kernighan neighbourhood structure can be considered as the champion among local search algorithms for the TSP. Unfortunately, the following theorem holds.

Theorem 4.3 *No safe distributed neighbourhood structure $\mathcal{N}_P$ can exist, with partial solutions of identical size, that is epimorphic or isomorphic to $\mathcal{N}_{lk}$.*

Theorem 4.3 states that it is not possible to design a parallel local search algorithm with a linear speed-up, that finds solutions with the same average cost as a sequential Lin-Kernighan algorithm. We can, however, still use the Lin-Kernighan neighbourhood as local neighbourhood. In [9] we define a distributed neighbourhood structure $\mathcal{N}_{lk,B,P}$ that is epimorphic to $\mathcal{N}_{3,B}$. Further, an algorithm that uses $\mathcal{N}_{lk,B,P}$ finds solutions with at most the same average cost as the solutions found by an algorithm that uses $\mathcal{N}'_{3,B,P}$.

An important topic is how the quality of solutions found by a parallel algorithm that uses $\mathcal{N}_{lk,B,P}$ compares to that of the solutions found by a sequential Lin-Kernighan algorithm. The total number of tours examined to verify global local optimality of $\mathcal{N}_{lk,B,P}$ is $\mathcal{O}((B+1) * \frac{N^4}{P^2})$. We see that this number decreases, with increasing P. The effect of this on the average cost of solutions found by a parallel algorithm that uses $\mathcal{N}_{lk,B,P}$ is revealed by an empirical study (see Section 4).

To conclude this section we briefly discuss the complexity of parallel local search algorithms that use the above neighbourhood structures. Since it is not possible to give a non-trivial upper bound on the number of iterations needed to find a global local optimum, we restrict ourselves to investigating the complexity to verify global local optimality of a solution.

The worst-case complexity to verify global local optimality of $\mathcal{N}_{2,P}$ is $\mathcal{O}(N * (\frac{2*N}{P} + P))$. The worst-case complexity to verify global local optimality of $\mathcal{N}_{3,B,P}$ is $\mathcal{O}(P + N * (\frac{B*N}{P} + B^2 + B * P))$, and for $\mathcal{N}'_{3,B,P}$ it is $\mathcal{O}(P + N * ((\frac{N}{P})^2 + B * \frac{N}{P} + B * P))$. Partially this is super-linear decreasing, with increasing P. The worst-case complexity to verify global local optimality of $\mathcal{N}_{lk,B,P}$ is $\mathcal{O}(P + N * ((B+1) * (\frac{N}{P})^3 + B * P))$, which is partially super-linear decreasing, with increasing P. If the complexity to verify global local optimality is an indication for the overall average-case complexity, we may expect a more than linear

Table 1
Computational results for ATT532 and PR1002 with $\mathcal{N}_{2,P}$.

P	f_{av}	$\Delta\%$	T(s)	EF	P	f_{av}	$\Delta\%$	T(s)	EF
1	29437	7.42	43.70	1	1	277779	7.23	753.0	1
10	29867	7.88	5.85	0.75	10	278226	7.40	91.4	0.82
30	29844	7.80	2.91	0.50	25	278822	7.63	53.3	0.57
50	29839	7.78	2.41	0.36	50	278139	7.37	30.7	0.49

speed-up for $\mathcal{N}'_{3,B,P}$ and $\mathcal{N}_{lk,B,P}$, possibly at the cost of a decrease in quality of the final solution.

5. Numerical results

The parallel local search algorithms sketched in the previous section, have been executed on networks of 50 and 400 INMOS T800 transputers. Starting tours have been obtained using the tour construction heuristic "Nearest Neighbour"[3]. We used instances with several thousands of cities obtained from the TSP library of [10]. Aspects regarding speed-up and efficiency, the influence of the number of processors on the average cost of the final solutions, and the choice for the value of B are discussed on the basis of the following (solved) instances: ATT532 (27686), PR1002 (259045). The number in the name of an instance denotes the number of cities of this instance, and the number in parentheses gives the cost of the optimal solution, or the best known lower bound.

In the tables below we give some of the computational results that we obtained. In these tables, P is the number of processors, f_{av} is the average cost of the final solution computed from 10 runs for ATT532, and from 5 runs for the other instances. The average excess over the optimal cost or its lower bound measured in percentages, is given by $\Delta\%$. The average run-time is given by T. The efficiency, defined as speed-up divided by P, is denoted by EF. In our definition of speed-up it is not required that a parallel algorithm produces the same output for each P.

5.1. Results for $\mathcal{N}_{2,P}$

From Table 1, we observe that the number of processes has indeed no influence on the average cost of the final solution, and that the efficiency is more than 50 percent for sufficiently large values of $N\mathbf{div}P$, the size of a partial solution, which is considered acceptable.

5.2. Results for $\mathcal{N}'_{3,B,P}$

From Table 2 we observe that, if we increase the number of processors, the run-time decreases more than linear, although there is only a small increase in the average cost of the final solution. This leads to the conclusion that tours obtained by 3-exchanges that are not generated in $\mathcal{N}'_{3,B,P}$ if the number of processors increases, generally do not have lower cost. These tours consist of all tours that are obtained by 3-exchanges in which the shortest sub-path has length more than $N\mathbf{div}P$ and some tours for which this length is more than B.

Table 2
Computational results for ATT532 and PR1002 with $\mathcal{N}'_{3,B,P}$

P	B	f_{av}	$\Delta\%$	T(s)	EF	P	B	f_{av}	$\Delta\%$	T(m)	EF
1	-	28565	3.17	16737	1	1	-	272965	5.37	10206	1
10	5	28482	2.88	681	2.48	25	4	268980	3.84	97.0	4.21
30	5	28537	3.07	113	4.94	50	4	268575	3.68	25.9	7.90
50	5	28574	3.21	49	6.83	200	4	269383	3.99	3.6	14.3

Table 3
Computational results for ATT532 and PR1002 with $\mathcal{N}_{3,B,P}$.

P	B	f_{av}	$\Delta\%$	T(s)	EF	P	B	f_{av}	$\Delta\%$	T(m)	EF
1	5	28612	3.35	831	1	1	4	269282	3.95	165.8	1
30	5	28604	3.32	51	0.54	25	4	269458	4.02	11.2	0.59
50	5	28622	3.38	33	0.50	50	4	269323	3.97	5.4	0.61

5.3. Results for $\mathcal{N}_{3,B,P}$

Table 3 shows that we obtain an acceptable efficiency, and that the number of processes has no influence on the average cost of the final solution.

If we compare the results obtained with $\mathcal{N}'_{3,B,P}$ with those obtained with $\mathcal{N}_{3,B,P}$, we observe that using $\mathcal{N}_{3,B,P}$ leads to considerable shorter run-times, at the cost of a small increase in the average cost of the final solution. The difference in run-time becomes even more significant at increasing difference between $N\mathbf{div}P$ and B.

5.4. Results for $\mathcal{N}_{lk,B,P}$

From Table 4, we observe that a more than linear speed-up is obtained, but at the cost of an increase of the average cost of the final solution.

If we compare the results we obtained with $\mathcal{N}_{lk,B,P}$ to the results obtained with $\mathcal{N}'_{3,B,P}$, we observe that solutions found by using $\mathcal{N}_{lk,B,P}$, have lower average cost than the solutions found by $\mathcal{N}'_{3,B,P}$, especially at larger values of $N\mathbf{div}P$. This is explained from the fact that the solution space can be more effectively searched in $\mathcal{N}_{lk,B,P}$ at larger values of $N\mathbf{div}P$. This also applies to the run-time of $\mathcal{N}_{lk,B,P}$ compared to the run-time of $\mathcal{N}'_{3,B,P}$, for at larger values of $N\mathbf{div}P$ the run-time required by $\mathcal{N}_{lk,B,P}$ is less than that required by $\mathcal{N}'_{3,B,P}$, in spite of the greater complexity of $\mathcal{N}_{lk,B,P}$.

Table 4
Computational results for ATT532 and PR1002 with $\mathcal{N}_{lk,B,P}$.

P	B	f_{av}	$\Delta\%$	T(s)	EF	P	B	f_{av}	$\Delta\%$	T(m)	EF
1	-	28139	1.64	7920	1	1	-	264366	2.05	5220	1
5	5	28274	2.12	689	1.55	25	4	268305	3.57	24.6	8.35
30	5	28503	2.95	163	1.63	50	4	267944	3.44	11.4	9.16
50	5	28555	3.14	108	1.47	50	19	266882	3.02	33.2	3.16

Table 5
Computational results for U2152 (64163) and U2319 (234256).

$\mathcal{N}$	P	B	f_{av}	$\Delta\%$	T(m)	$\mathcal{N}$	P	B	f_{av}	$\Delta\%$	T(m)
$\mathcal{N}_{2,P}$	50	-	69724	8.67	1.7	$\mathcal{N}_{2,P}$	50	-	245800	4.93	2.9
$\mathcal{N}_{3,B,P}$	50	10	66954	4.35	90.5	$\mathcal{N}_{3,B,P}$	50	4	240318	2.59	27.4
$\mathcal{N}'_{3,B,P}$	50	4	67090	4.56	261.8	$\mathcal{N}'_{3,B,P}$	50	4	239645	2.30	353.2
$\mathcal{N}_{lk,B,P}$	50	4	67003	4.43	74.0	$\mathcal{N}_{lk,B,P}$	50	4	237209	1.26	71.4

Table 6
Computational results for PR2392 (378032) and PCB3038 (137617).

$\mathcal{N}$	P	B	f_{av}	$\Delta\%$	T(m)	$\mathcal{N}$	P	B	f_{av}	$\Delta\%$	T(m)
$\mathcal{N}_{3,B,P}$	50	4	395318	4.57	39.2	$\mathcal{N}_{2,P}$	50	-	149425	8.58	5.7
$\mathcal{N}'_{3,B,P}$	50	4	393160	4.00	392.0	$\mathcal{N}_{3,B,P}$	50	10	143386	4.19	241.3
$\mathcal{N}'_{3,B,P}$	400	4	394805	4.43	11.6	$\mathcal{N}'_{3,B,P}$	50	4	143449	4.24	915.3
$\mathcal{N}_{lk,B,P}$	50	4	390465	3.29	104.0	$\mathcal{N}_{lk,B,P}$	50	4	142572	3.60	160.0
$\mathcal{N}_{lk,B,P}$	400	4	394276	4.30	11.3	$\mathcal{N}_{lk,B,P}$	400	4	143609	4.35	30.2

5.5. Results for other TSP instances

Tables 5 and 6 show that for sufficiently large B, $\mathcal{N}_{3,B,P}$ gives the same quality solutions as $\mathcal{N}'_{3,B,P}$ in shorter time, and that the conclusions from the previous sections apply to these instances as well.

5.6. Summary

We have discussed the performance of parallel local search algorithms for the TSP using the distributed neighbourhood structures $\mathcal{N}_{2,P}$, $\mathcal{N}'_{3,B,P}$, $\mathcal{N}_{3,B,P}$, and $\mathcal{N}_{lk,B,P}$. From the results we conclude that we obtain an acceptable efficiency for $\mathcal{N}_{2,P}$ and $\mathcal{N}_{3,B,P}$, and that the average cost of the final solutions does not depend on the number of processes for these neighbourhood structures. For the neighbourhood structures $\mathcal{N}'_{3,B,P}$ and $\mathcal{N}_{lk,B,P}$ we obtain a more than linear speed-up, with no significant increase in the cost of the final solutions for $\mathcal{N}'_{3,B,P}$, but at the cost of a decrease in quality of the final solutions for $\mathcal{N}_{lk,B,P}$. Furthermore, we conclude that small values for B, up to roughly 10, can be used, and that $\mathcal{N}_{3,B,P}$ gives roughly equal quality solutions as $\mathcal{N}'_{3,B,P}$ in shorter time, by taking a larger value for B. Finally, we conclude that for TSP instances with N cities and P processes such that the value of $N\,\mathbf{div}\,P$ is sufficiently large, $\mathcal{N}_{lk,B,P}$ finds better solutions in shorter time than $\mathcal{N}_{3,B,P}$ and $\mathcal{N}'_{3,B,P}$. A conservative estimation for this value of $N\,\mathbf{div}\,P$ is roughly 40. For smaller partial solution sizes, $\mathcal{N}_{lk,B,P}$ gives results that are comparable with the results obtained by $\mathcal{N}_{3,B,P}$ and $\mathcal{N}'_{3,B,P}$. So, for sufficiently large instances, $\mathcal{N}_{lk,B,P}$ gives the best results with respect to both run-time and quality of the final solutions, although we cannot obtain the same quality solutions as a sequential algorithm that uses $\mathcal{N}_{lk}$.

References

1 C.H. Papadimitriou and K. Steiglitz [1982], *Combinatorial Optimization: Algorithms and Complexity*, Prentice Hall, New York.

2 M.R. Garey and D.S. Johnson [1979], *Computers and Intractability, A Guide to the Theory of NP-Completeness*, W.H. Freeman and Co., San Francisco.

3 E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys (eds.) [1985], *The Travelling Salesman Problem*, John Wiley and Sons, Chichester.

4 D.S. Johnson [1990], Local Optimization and the Traveling Salesman Problem, *Proc. 17th Colloquium on Automata, Languages and Programming (ICALP '90)*, Springer Verlag, LNCS 447, 446-461.

5 S. Lin [1965], Computer Solutions of the Travelling Salesman Problem, *Bell Syst. Tech. J. 44*, 2245-2269.

6 S. Lin and B.W. Kernighan [1973], An Effective Heuristic Algorithm for the Traveling Salesman Problem, *Oper. Res. 21*, 498-516.

7 I. Or [1976], *Travelling Salesman Type Combinatorial Problems and their Relation to the Logistics of Regional Blood Banking*, Ph.D. thesis, Northwestern University, Evanston.

8 E. Aarts and J. Korst [1989], *Simulated Annealing and Boltzmann Machines*, John Wiley and Sons, Chichester.

9 M.G.A. Verhoeven and E.H.L. Aarts [1992], Parallel Local Search and the Travelling Salesman Problem, in preparation.

10 G. Reinelt [1990], TSPLIB: A Travelling Salesman Problem Library, *Report no. 250 Schwerpunktprogramm der Deutschen Forschungsgemeinschaft*, Universität Augsburg.