数理モデル化と問題解決 18－5
（1998. 3. 20）

タブー探索の拡張と配送問題への適用
— 局所最適解で分岐探索する Taboo Tree アルゴリズム —

上田晴康
富士通研究所（株）マルチメディアシステム研究所
情報科学研究部

局所最適解毎にタブー探索を行なう Taboo Tree アルゴリズムを提案する。このアルゴリズムでは局所最適解でループが検出できるため比較的短いタブーリストで幅広い探索を行なうことができる。タブーリスト長の調整もほとんど必要ない。このアルゴリズムでは局所最適解を単位として枝別れしながら探索を行うことができるため、集中的に解を探索することができるが、bounded depth first search の技法を適用することで多様性を増やすこともできる。ベンチマークテストで実験を行ない探索性能を評価した。

An Extension of Taboo Search and Application to Vehicle Routing
–Taboo Tree Algorithm Branching at Local Optimum Solution—

Haruyasu Ueda
Information Science Laboratory,
Multimedia Systems Laboratories,
FUJITSU LABORATORIES LTD.

An extension with taboo search named Taboo Tree algorithm is proposed. Taboo Tree algorithm keeps several local optimum solutions and their taboo lists. Therefore, it can continue or restart taboo search whenever a local optimum solution is found. As it finds all significant loop at the local optimum solution, it needs just short taboo list. It can search very intensively when it always restarts from the best solution. It can also be diversified by applying bounded depth first search to selecting restarting solution. It is evaluated with a set of benchmark tests.

1 はじめに

配送計画問題 (Vehicle Routing Problem) は、多くの研究者によって最適解を探索する方法が研究されてきた [16, 6, 2]。組み合わせ最適化に対するさまざまな手法が試みられ、さまざまなヒューリスティクスを用いた探索 [15, 13, 7] に加え Genetic Algorithm[10], タブー探索 [11, 3] 及びそれらの組み合わせによる成果 [17] が報告されている。

タブー探索に関しては高速でかつ良い解が得られることが知られているが、タブーリスト長というパラメータがあって、これによって大きく性能が変わるためパラメータの決定が問題となっている [5]。タブーリスト長の自動決定に関する研究も行われているが [1]、理論的な解析はまだ行なわれていない。そこで、本論分ではタブー探索の局所最適解を用いてループ検出する拡張をした Taboo Tree アルゴリズムを提案する。このアルゴリズムでは局所最適解から探索を枝分かれさせて探索を行なうことができるので、従来のタブー探索に比べて良い解の周辺を集中的に探索することができる。

本論文では、配送計画問題について述べた後、アルゴリズムの説明を行ない、性能を示すため VRP のための Solomon によるベンチマーク [15] での実験結果を示す。

2 配送計画問題 (VRP)

配送計画問題は、単一の配送センター (S_0)、配送先 $S_i(i = 1..n)$ とその配送量 (Ws_i)、配送車 $V_j(j = 1..m)$ とその最大積載量 Wv_j、配送センター及び配送先との距離 ($D_{i,j}(i, j = 0..n)$) が与えられたときに各配送車の積載量が最大積載量を超えないように配送車に配送先を割り当てかつ、使用配送車の台数の最小化と総配送距離の最小化を行うものである。$V_j(m_j$ 個所の配送先を回る）の k 番目の配送先を S_{jk}

$(k < m_j)$ とおくと、

$$Wv_j > \sum_k Ws_{jk}$$

$$\text{minimize} \sum_j \begin{cases} 1 & V_j\text{が配送するとき} \\ 0 & V_j\text{が配送しないとき} \end{cases}$$

$$\text{minimize} \sum_{j,k} D_{S_{j(k-1)},S_{jk}}$$

さらに、この論文では各配送先に到着時間制約 (Time Window) 及び配送車の帰着時間制約のある VRPTW を対象とする。到着時間制約は配送先 i での到着時間がある一定の時間枠 $[Topen_i, Tclose_i]$ の間になくてはならないというものであり、帰着時間制約はすべての配送車は $Tclose_0$ までに帰着しなくてはならないというものである。各配送先では積み下ろし時間 P_i がかかり、$Topen_i$ 以前に配送先 S_i に到着した場合 $Topen_i$ までそこで待つものとする。また、距離は時間距離であらわされているものとする。これは以下のように定式化される。

$$\begin{aligned} Arrival_{jk} &= Departure_{j(k-1)} + D_{S_{j(k-1)},S_{jk}} \\ Departure_{jk} &= max(Arrival_{jk}, Topen_{jk}) + P_{jk} \\ Departure_{*0} &= 0 \\ Arrival_{j(m_j+1)} &\leq Tclose_0 \end{aligned}$$

(ただし、P_{jk} などは $P_{S_{jk}}$ の省略形)

2.1 探索範囲

探索範囲は大まかに制約を満たした解 (feasible solution) の範囲と制約違反をしている解 (infeasible solution) の 2 種類に分けられる。制約違反に関してペナルティを課すことで制約違反の解の範囲を探索するやり方もあるが 第一に一度制約を違反してしまうと制約を満たすようにするのが難しいこと、特に最後に選られた解が制約違反の解の場合に必ずしも最適解を選られないこと、第二にペナルティの値を動的に管理しないとうまく探索が行われないことの 2 点の理由により本論文では制約を満たした解のみの範囲で探索を行う。

2.2 目的関数

本論文で扱う目的関数は配送車最小の解の中で配送距離を最小にするというものである。探索の方向を誘導するために以下のような目的関数を設定し、その最小化をすることで解を探索した。

$Cost = a$·使用配送車数$+b$·総距離$+c$·帰着時間総和$+d$· 最短帰着時間

但し

a, b, c および d は 設定可能なパラメータ

$$\begin{aligned} \text{総距離} &= \sum_{j,k} D_{S_{j(k_1)},S_{jk}} \\ \text{帰着時間総和} &= \sum_j Arrival_{j(m_j+1)} \\ \text{最短帰着時間} &= \min_j Arrival_{j(m_j+1)} \end{aligned}$$

3 アルゴリズム

3.1 Taboo 探索

タブー探索 [5] は山登り法や Simulated Annealing と同じローカルサーチの一種であり、一つの解をもとに適当なヒューリスティクス (move) を順に作用させてより良い解を探索する。しかし、単純な山登り法と異なり、局所最適解におちいっても探索を継続することができる。

タブー探索では局所最適解から抜け出す際にループを陥らないためにタブーリストと呼ばれる情報を使う。直感的にはタブーリストには最近行われた探索の履歴が記録され、局所解から抜け出す際にはその履歴を繰り返さない範囲で目的関数を最小にする方向へ探索を行う。このため、峠を越えて局所最適解から抜け出すことができる。

3.1.1 タブーリスト

タブーリスト構造として TSP および VRP で典型的に用いられている行列表現を用いている [5]。この行列 $Taboo_{i,j}$ は配送先 S_i から S_j へ向かう経路を表現しており、その値が現在の繰り返し回数よりも大きい時タブーとする。

加えられるタブーは配送ルート中に新たに追加された経路である。例えば $S_1 \rightarrow S_2$ という経路が $S_1 \rightarrow S_3 \rightarrow S_2$ という経路に変わった場合 $Taboo_{1,3}$ と $Taboo_{3,2}$ のがタブーとなり、値を 現在の繰り返し回数 + タブーリスト長 に変える。

探索の際には解を変形する際に除去される経路が一つでもタブーに該当すればそのヒューリスティクスはタブーとなる。先の例の後では $S_1 \rightarrow S_3$ または $S_3 \rightarrow S_2$ の経路のいずれかを除去するような解の変形はすべてタブーとなる。

3.1.2 VRP のためのローカルサーチ

ローカルサーチではヒューリスティクス (move) の種類によって探索効率が大きく異なる。本論文では以下のヒューリスティクスを用いている。

- 2-OPT*: 二つの配送車の配送順を途中で切り替える。[12]

- OR-OPT: 一つの配送順の中で引き続いたいくつかの配送先を前後に移動する。探索範囲を狭めるため移動する配送先は3個所まで。[8, 12]
- INSERT: 二つの配送車の片方から一つ配送先を取り出して、もう一方に挿入する。挿入位置は最適の位置を計算する。
- SWAP: 二つの配送車の配送先を一つづつ取り出して交換する。挿入される位置は最適の位置をそれぞれ選ぶ。
- JOIN: 一つの配送車の使用を止め、その配送先を適当に他の配送車に割り振る。探索範囲を狭めるために6以上の配送先のある配送車は探索を行わない。

本論文ではすべてのヒューリスティクスに関して可能な組み合わせをすべて試し、適用した結果が制約違反の解にならず、そのヒューリスティクスがタブーでもない組み合わせのうち、適用した結果の解の評価関数値がもっとも小さいものを一つ選んで適用する。

3.1.3 初期解

タブー探索では既知の解を改良するため初期解を別途作成するアルゴリズムが必要である。本論文ではSweep ヒューリスティクス [4] の第一ステップを用いた。これは配送センターを中心とした角度の順に貪欲に配送先を配送車に挿入して行くアルゴリズムである。配送車の中の配送順序は挿入する配送先毎に最適の（帰着時刻がもっとも早くなる）位置を計算する。このSweepヒューリスティクスはパラメータとして最初に挿入する配送先を必要とするので、初期解では乱数用いてパラメータとして与えた。

3.1.4 終了条件

タブー探索では原理的にはいつまででも探索を行うことが可能であるため何らかの終了条件を入れる必要がある。本論文では単純にヒューリスティクスの適用回数をパラメータとして与えることにし、その回数以内で見つかったもっとも良い解を出力するものとする。

3.2 Taboo Tree アルゴリズム

タブー探索ではタブーリスト長と呼ばれるパラメータの決定が大きな問題としてあげられている [5]。タブーリスト長が短すぎるとループを起こしやすくなり、長すぎると最適解の近くまできているのに到達できなくなる可能性があるためである。

そこで、ループを直接検出するTaboo Tree アルゴリズム（図1）を提案する。Taboo Tree アルゴリズムは、ループが起きるときには必ず同じ局所最適解を経由することを利用して、探索のすべての解を記憶する代わりに局所最適解のみを記憶して探索を行う。このことにより、タブーリスト長はごく短いものですみ、ほとんど調整する必要がなくなる。各局所最適解では、解自身とともに、その解から行われた探索方向をタブーリストとして記憶しておく。これにより、次にその局所最適解から探索を継続したときに同じ経路を探索することを防ぐ。記憶されたタブーリストは繰り返し数=0の時の値になるように正規化しておく。もしもある局所最適解でループが検出された場合は、繰り返し数を0にリセットして、記憶されたタブーリストを用いて探索を再開する。この拡張により、局所最適解から局所最適解を見つける過程が木状（正確には連結した有向グラフ）になるため、本アルゴリズムをTaboo Treeアルゴリズムと呼ぶ。

局所最適解から行われた探索方向をタブーリストとして保存するために、探索の過程では通常のタブーリストに加え、逆むきのタブーリストも更新をする。逆向きのタブーリストは最大の要素が最大タブー長になった行列でヒューリスティクスを一つ実行する毎に更新される値を小さくして行く。更新されるタブーも通常と逆に切断された経路をタブーとする。これにより、探索を進めた方向から局所最適解へ向かってきたのと同じ効果を持つタブーリストを作成できる。次の局所最適解に達した時点で逆むきのタブーリストを出発点の局所最適解のタブーリストとマージをする。タブーリストのマージでは同じタブーが二つのリストにあった場合、より大きいタブーの値を用いる。

局所最適解のタブーリストの各要素の値はその解から探索を再開（あるいは継続）するたびに単調に増加する。このため、何度も探索を行うことにより、いつかその局所最適解からの全てのヒューリスティクスがタブーになるようになる。このような解はもはや探索を再開することができない。そこで、このような局所最適解を成熟した (matured) 解と呼び、成熟していない局所最適解（訓練中 (apprentice) と呼ぶ）と分けて置く。もしも探索を行った結果成熟した解に行き着いてしまった場合は訓練中の適当な解から探索を再開する。

3.3 多様性と集中

Taboo Treeアルゴリズムではある局所最適解を見つけた後、どの局所最適解から探索を継続するかはまったく自由である。すなわち図1のアルゴリズ

```
Taboo_tree(initial_solution) returns solution
 parameter
  int max_iteration;
  int save_size;
 variable
  solution: local_solution, best_solution;
  taboo_list: local_taboo, reverse_taboo;
  array[0..save_size] of
    {solution, taboo}: apprentice;
  array[0..save_size] of solution: matured;
  int iteration;

 apprentice = φ;
 matured = φ;
 best_solution = initial_solution;
 apprentice[0].solution = initial_solution;
 last_index = 0;

 for (iteration = 0 ..max_iteration) {
  if (last_index == -1)
   last_index = 0;
  selected = last_index;
  local_solution =
    apprentice[selected].solution;
  local_taboo =
    apprentice[selected].taboo_list;
  iterations =
    taboo_search_until_local_min(
      local_solution,local_taboo,reverse_taboo);
  if (local_solution.cost<best_solution.cost)
   best_solution = local_solution;
  merge_reverse_taboo(
    apprentice[selected].taboo,reverse_taboo);
  last_index = -1;
  if (taboo_search successed) {
   if (local_solution ∉ matured &&
       local_solution ∉ apprentice) {
    if (apprentice has save_max solutions &&
        worst solution != local_solution)
      worst solution is deleted;
    insert {local_solution, new taboo_list}
        into apprentice;
    sort(apprentice);
    last_index = inserted position;
   } else if (local_solution ∈ apprentice)
    last_index = index in apprentice;
  }else {  /* taboo search failed */
   /* It becomes matured */
   if ( local_solution ∉ matured) {
    insert apprentice[selected].solution
        into matured;
    sort (matured);
    remove apprentice[selected] from apprentice;
   }
  }
 }
return best_solution;
```

図 1: Taboo Tree アルゴリズム

ムでは変数 selected を適当に設定することができる。

従来のタブー探索を行うには常に最後の局所最適解から探索を継続すれば良い。図 1はこれを行っている。これに対して、常に局所最適解の内もっとも良い解から探索を行えば集中的に最適解近傍を探索することができる。

一方、一般的に集中しつつもある程度の多様性を持たせた探索が大域的な最適解を見つけられるとされている [5]。そこで、本論文では多様性を確保するために bounded depth first の技法を Taboo Tree アルゴリズムに適用したものを提案する。すなわち、ある一定の回数は通常のタブー探索を行い、その回数以内にそれまでのもっとも良い局所最適解より良い解を探索できなかった場合に、もっとも良い局所最適解より探索を再開するようにする (図 2)。

```
Bounded_Depth_First_Taboo_tree(initial_solution)
    returns solution
  parameter
    int max_iterations;
    int save_size;
    int bound;
  variable
    int depth
    ...
  best_solution = initial_solution;
  apprentice[0].solution = initial_solution;
  last_index = 0;
  depth = bound;
  for (iteration = 0 ..max_iterations) {
    if (last_index is not known)
       last_index = 0;
    if (last_index > 0) {
      depth = depth - 1;
      if (depth < 0)
        last_index = 0;
    }
    if (last_index == 0)
      deptha = bound;
    selected = last_index;
    以下図 1に同じ
```

図 2: Bounded Depth First Taboo Tree アルゴリズム

また、訓練中の局所最適解がほとんど同じ解で占められてしまう効果の影響を減らすために、局所最適解をクリアするのが効果的である。このために Bounded Depth First Taboo Tree アルゴリズムで得られた最適解を次の Bounded Depth First Taboo Tree アルゴリズムの初期解として与えて実行することを繰り返した。繰り返し回数はパラメータとして与えた。

4 実験結果

Solomon のベンチマークテスト [15] を用いて Bounded Depth First Taboo Tree アルゴリズムの性能の評価を行った。パラメータの値はすべて経験的に決められた (表 1)。実験はそれぞれのベンチマークテストに関して 1 回ずつ行った結果である。

表 2、3はそれぞれの問題に対して Best Known の結果 [14] と本論文で選られた結果に関して配送車数と総配送距離を示したものである。また、配送距離が相対的にどの程度増えているかも示してある。配送車数が best known と同じにならなかったものは台数に - マークがついている。

ここに示されるように Taboo Tree アルゴリズム

評価関数の配送車数の係数	10,000
評価関数の総距離の係数	0.07
評価関数の帰着時間総和の係数	0.01
評価関数の最短帰着時間の係数	0.02
タブーリスト長	10
ヒューリスティクスの適用回数上限	1,000
局所最適解の保存数	128
一回の引き続く探索の深さの上限	15
B.D.F. Taboo Tree の繰り返し回数	5

表 1: 用いたパラメータ

問題番号	Best Known		Taboo Tree		
	車数	総距離	車数	総距離	増分 %
R101	18	1608[a]	-19	1657.48	3.1%
R102	17	1434[a]	-18	1485.08	3.6%
R103	13	1207[b]	-14	1227.23	1.7%
R104	10	982.01[f]	-11	1023.41	4.2%
R105	14	1377.11[f]	-15	1398.06	1.5%
R106	12	1252.03[f]	-13	1262.90	0.9%
R107	10	1159.86[f]	-11	1082.75	-6.6%
R108	9	980.95[f]	-10	980.31	-0.1%
R109	11	1235.68[f]	-12	1211.17	-2.0%
R110	10	1080.36[f]	-11	1154.81	6.9%
R111	10	1129.88[f]	-12	1151.01	1.9%
R112	10	953.63[f]	10	993.52	4.2%
C101	10	827[a]	10	829.01	0.2%
C102	10	827[a]	10	829.01	0.2%
C103	10	828.06[f]	10	829.01	0.1%
C104	10	840[b]	10	908.54	8.2%
C105	10	828.94[f]	10	829.01	0.0%
C106	10	827[a]	10	829.01	0.2%
C107	10	829[e]	10	829.01	0.0%
C108	10	828.94[f]	10	829.01	0.2%
C109	10	828.94[f]	10	845.14	1.9%
RC101	14	1669[b]	-15	1665.30	-0.2%
RC102	13	1477.54[f]	-14	1480.68	0.2%
RC103	11	1110[b]	-12	1342.27	20.9%
RC104	10	1135.83[f]	10	1168.73	2.9%
RC105	13	1733.56[f]	-15	1575.11	-9.1%
RC106	12	1384.92[f]	-13	1401.11	1.2%
RC107	11	1230.95[f]	-12	1274.26	3.5%
RC108	10	1170.70[f]	-11	1161.07	-0.8%

表 2: 実験結果 (1)

は最適解には及ばないが、かなり良い結果を出していると考えられる。配送先がクラスタ状になっている c1, c2 の二つの問題群に関してはほとんどの問題で総距離を best known から 5% 以内まで減らすことができた。

問題番号	Best Known		Taboo Tree		
	車数	総距離	車数	総距離	増分 %
R201	4	1281[f]	4	1296.71	1.2%
R202	3	1530.49[c]	-4	1162.52	-1.1%
R203	3	948.74[f]	3	1041.39	9.8%
R204	2	869.29[f]	-3	839.42	4.6%
R205	3	1063.24[f]	3	1112.39	4.6%
R206	3	833[b]	3	997.21	19.7%
R207	3	814.78[f]	3	960.56	17.9%
R208	2	738.60[f]	2	855.78	15.9%
R209	2	855[b]	-3	1031.92	20.7%
R210	3	967.50[f]	3	1047.88	8.3%
R211	2	949.50[f]	-3	915.04	-3.6%
C201	3	590[e]	3	591.58	0.3%
C202	3	591[e]	3	613.04	3.7%
C203	3	591.17[f]	3	677.98	14.7%
C204	3	590.60[c]	3	687.85	16.5%
C205	3	588.88[f]	3	588.90	0.0%
C206	3	588[e]	3	588.52	0.1%
C207	3	588[e]	3	588.35	0.1%
C208	3	588[e]	3	588.52	0.1%
RC201	4	1249[b]	4	1532.67	22.7%
RC202	4	1165.57[f]	4	1320.24	13.3%
RC203	3	1079,57[f]	3	1132.20	4.9%
RC204	3	806.75[f]	3	913.02	13.2%
RC205	4	1333.71[f]	4	1397.92	4.8%
RC206	3	1212.64[f]	-4	1214.78	0.2%
RC207	3	1085.61[f]	-4	1093.19	0.7%
RC208	3	833.97[f]	3	950.08	13.9%

表 3: 実験結果 (2)

a. Desrochers et al. (1992)[2]
b. Thangiah et al. (1994)[17]
c. Potvin & Bengio (1994)[9]
d. Thompson & Psaraftis (1993)[18]
e. Potvin et al. (1993)[13]
f. Rochat et al. (1995)[14]

一方配送先が広範囲に渡っている r1, r2, rc1, rc2 の問題群では配送車数が best known と同じで総距離を 10% 以内にできる問題は数個に限られている。また、初期解の影響もかなりあり、広い意味での局所最適解に捕まってしまっているといえる。

この原因として実行の経過から観察されたのは、訓練中の局所最適解が少しずつ異なった解で占められてしまうことである。ある程度広い最適解では非常に多くの局所最適解が存在して、でこぼこの盆地状の探索空間となってしまうので局所解の保存数を増やしても効果はほとんどない。また、一度に行う探索の深さを増やすと広い範囲を探しに行くが、選られる解のうち、良い局所最適解はきわめて似てしまうため一定の効果しかない。

R1	C1	RC1	R2	C2	RC2
2210.9	1977.3	1723.8	7323.9	2592.2	6189.5

表 4: 計算時間

4.1 計算時間

現在のプロトタイプでの実行時間を Ultra Sparc (250MHz) を用いて調べた。実行は各問題を 1 回ずつ行ない、消費した CPU 時間 (user time) を問題群毎に集計して平均をしてものを示した (表 4)。単位は秒である。

この結果から分かるようにかなり計算時間がかかっている。これは 3.1.2 で示したように、すべてのヒューリスティクスで可能なすべての組合せを試しているためと、ヒューリスティクスの適用を一度する毎にすべての探索をやり直しているためである。また、多様性を増やすために何度も Bounded Depth Taboo Tree を繰り返しているのも大きな原因となっている。

各ヒューリスティクスで、変更を行なう配送先同士の距離が近い場合に限定して探索する (h-neighbor 探索 [12]) ことと、各探索において探索結果を保存し、不必要な再計算を省略することで大幅な高速化が行なわれるものと思われる。

また、解の質とも関係するが、探索の過程で局所最適解の多様性をより広げることで探索アルゴリズムの繰り返しを減らすことによっても計算時間を減らせるものと思われる。

5 まとめ

本論分ではタブー探索の局所最適解を用いてループ検出をするように拡張した Taboo Tree アルゴリズムを提案した。また、多様性を増やすための改良として bounded depth first search の技法を適用し、時々得られた局所最適解をクリアすることでよりよい解を探索できることが分かったが、ベンチマークテストではいくつかの問題群で、より多様性の高い探索をする必要があることが示された。

参考文献

[1] R. Battiti and G. Tecchiolli. The reactive tabu search. *ORSA J. on Computing*, 6(2), 1994.

[2] M. Desrochers, J. Desrosiers, and M. Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Op. Res.*, 40(2), 1992.

[3] M. Gendreau, A Hertz, and G. Laporte. A tabu search heuristic for the vehicle routing problem. *Management Science*, 40(10), 1994.

[4] Billy E. Gillett and Leland R. Miller. A heuristic algorithm for the vehicle-dispatch problem. *Op. Res.*, 22(2), 1974.

[5] F. Glover, E. Taillard, and D. de Werra. A user's guide to tabu search. *Annals of Op. Res.*, 41, 1993.

[6] B Golden and A. Assad(eds.). *Vehicle Routing with TIme Window Constraings: Algorithmic Solutions.* American Science Press, 1986.

[7] Y. W. Kokosidis, B. Powell, and M. M. Slolomon. An optimization based heuristic for vehicle routing and scheduling with time window constraints. *Transportation Science*, 26(2), 1992.

[8] I. Or. Traveling salesman-type combinational problems and their relation to the logistics of blood banking, 1976.

[9] J.-Y. Potvin and S. Bengio. A genetic approach to the vehicle routing problem with time windows. Tech. Rep. CRT-953, Centre de recherche sur les transports, Univ. de Montreal, 1994.

[10] J.-Y. Potvin and S. Bengjo. A genetic approach to the vehicle routing problem with time windows. Tehcnical Report CRT-953, Centre de Recherche sur les Transports, Universite de Montreal, 1993.

[11] J.-Y. Potvin, T. Kervahut, B.-L. Garcia, and J.-M. Rousseau. A tabu search heuristic for the vehicle routing problem with time windows. Tech. Rep. CRT-855, Centre de Recherche sur les Transports, Universite de Montreal, 1992.

[12] J.-Y. Potvin, Tanguy Kervahut, Bruno L. Garcia, and J.-M. Rousseau. The vehicle routing problem with time windows part I: Tabu search. *INFORMS J. on Computing*, 8(2), 1996.

[13] J.-Y. Potvin and J.-M. Rousseau. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European J. of Op. Res.*, 66, 1993.

[14] Y. Rochat and E.D. Taillard. Probabilistic diversification and intensification in local search for vehicle routing. *J. of Heuristics*, 1(1), 1995.

[15] M. M. Solomon. Algorithms for the vehicle routing and scheduling problem with time window constraints. *Op. Res.*, 35(2), 1987.

[16] M. M. Solomon and J. Desrosiers. Time window constrained routing and scheduling problems: A survey. *Transportation Science*, 22(1), 86.

[17] S. R. Thangiah, I. H. Osman, and T. Sun. Hybrid genetic algorithm, simulated annealing and tabu search methods for vehicle routing problems with time windows. Technical Report UKC/OR94/4, Inst. of Math. and Stat., U. of Kent, 1995.

[18] P. M. Thompson and H. Psaraftis. Cyclic transfer algorithms for multi-vehicle routing and scheduling problems. *Op. Res.*, 41, 1989.