A Compressed Annealing Heuristic for the Traveling Salesman Problem with Time Windows

Jeffrey W. Ohlmann • Barrett W. Thomas
Department of Management Sciences, University of Iowa
108 John Pappajohn Business Building, Iowa City, Iowa 52242-1000
jeffrey-ohlmann@uiowa.edu • barrett-thomas@uiowa.edu

This paper describes a variant of simulated annealing incorporating a variable penalty method to solve the traveling salesman problem with time windows (TSPTW). Augmenting temperature from traditional simulated annealing with the concept of pressure (analogous to the value of the penalty multiplier), compressed annealing relaxes the time window constraints by integrating a penalty method within a stochastic search procedure. Computational results validate the value of a variable penalty method versus a static penalty approach. Compressed annealing compares favorably with benchmark results in the literature, obtaining best-known results for numerous instances.

Key words: traveling salesman; time windows; heuristics; simulated annealing; penalty methods

---

1. Introduction

With the production trends of "lean manufacturing" and "just-in-time" operations, an increased premium is placed on the freight industry to provide timely, efficient service. To address the resulting time-constrained routing problem, we apply an extension of simulated annealing to the traveling salesman problem with time windows (TSPTW). The TSPTW consists of finding a minimum-cost tour, starting from and returning to the same unique depot, that visits a set of customers exactly once, each of whom must be visited within a specific time window. Practical applications of the TSPTW abound in the industrial and service sectors: package delivery, bank couriers, busing logistics, and material handling systems with automated guided vehicles. In addition, the TSPTW is mathematically equivalent to time-sensitive production scheduling problems that are prevalent in manufacturing.

From the perspective of a fleet manager, the TSPTW is a sub-problem of the vehicle routing problem with time windows (VRPTW), in which a fleet of vehicles must be routed to satisfy a set of customers with time-sensitive demands. As part of "cluster first, route second" approaches to the VRPTW, TSPTW solution methods can be of great utility (Wolfler Calvo, 2000; Gendreau et al., 1998).

Solution approaches for the TSPTW range from exact mathematical programming techniques to various heuristic approaches. Exact approaches to the TSPTW have focused on integer and dynamic programming techniques. Christofides et al. (1981) and Baker (1983) present branch-and-bound algorithms that solve problems with up to 50 vertices, but require "moderately tight" time windows and/or little overlap between them. Langevin et al. (1993) introduce a two-commodity flow formulation well-suited to handling time windows; they solve instances with up to 40 nodes. Dumas et al. (1995) extend earlier dynamic programming approaches by using state space reduction techniques that enable the solution of problems up to 200 customers. In an alternate approach, Pesant et al. utilize constraint programming to develop an exact method (Pesant et al., 1998) and a heuristic (Pesant et al., 1999) for the TSPTW. Similarly, Focacci et al. (2002) embed optimization techniques within a constraint programming approach.

Because of limitations with exact formulations (Savelsbergh (1985) proves that even finding a feasible solution to the TSPTW is an NP-hard problem), there exists a facet of research focusing on heuristic techniques for the TSPTW. Carlton and Barnes (1996) solve the TSPTW with a tabu search approach that considers infeasible solutions in its search neighborhood through the implementation of a static penalty function. In contrast to the static penalty in Carlton and Barnes, the approach presented here uses a dynamic penalty. Gendreau et al. (1998) offer a construction and post-optimization heuristic based on a near-optimal TSP heuristic presented by Gendreau et al. (1992). Wolfler Calvo (2000) introduces a heuristic that first constructs an initial tour using a unique assignment relaxation and then improves upon this tour via local search. Such heuristic approaches to the TSPTW are particularly advantageous for large instances (> 200 customers) and instances with wide time windows.

The compressed annealing approach that we present in this paper has three main advantages over previously published solution approaches to the TSPTW. First, we emphasize the performance of the algorithm on benchmark sets of TSPTW problems. We obtain new best-known solutions for a number of instances and match the previously best-known solution in a majority of the remaining instances. As the first work to incorporate insight from the theoretical results in Ohlmann et al. (2004), we demonstrate that compressed annealing consistently converges to good solutions, and particularly exhibits potential for large instances with wide time windows. Second, implementations of compressed annealing do not require a commercial solver. For a transportation company that must solve TSPTW instances at a large number of sites, solution methods requiring commercial solvers are prohibitively expensive. Third, unlike most other heuristics for the TSPTW, compressed annealing is a general problem-solving method with known convergence results (Ohlmann et al., 2004). As an extension of the well-studied metaheuristic simulated annealing, compressed annealing also benefits from a broad body of literature and applications.

The remainder of this paper is outlined as follows. In §2, we present the modeling formulation and accompanying assumptions. We introduce compressed annealing in §3 and discuss its parameters along with a parameter calibration scheme. In §4, we report on compressed annealing's performance with regard to well-known test sets from the TSPTW literature. The benefit of a variable penalty approach is demonstrated

with comparisons to a static-penalty implementation of simulated annealing, while the overall effectiveness of compressed annealing is measured with respect to the best-known results from the TSPTW literature. We conclude the paper by summarizing our research and identifying areas for further study in §5.

2. Model Formulation

To formally define the TSPTW, let $G = (N, A)$ be a finite graph, where $N = \{0, 1, \ldots, n\}$ is the finite set of nodes or customers and $A = N \times N$ is the set of arcs connecting customers. We assume that there exists an arc $(i, j) \in A$ for every $i, j \in N$. A tour is defined by the order in which the n customers are visited and denoted by $\Im = \{p_0, p_1, \ldots, p_n, p_{n+1}\}$, where p_i denotes the index of the customer in the i^{th} position of the tour. Let customer 0 denote the depot and assume that every tour begins and ends at the depot, i.e., $p_0 = 0$ and $p_{n+1} = 0$. Each of the remaining n customers occupies one position ranging from p_1 to p_n inclusive.

For $j = 0, \ldots, n$, there is a cost, $c(a_j)$, for traversing the arc $a_j = (p_j, p_{j+1})$. This cost of traversing the arc between the j^{th} customer and the $(j+1)^{\text{th}}$ customer in the route generally consists of any service time at customer p_j plus the travel time from customer p_j to customer p_{j+1}. Associated with each customer i is time window, $[e_i, l_i]$, during which the customer i must be visited. We assume that waiting is permitted; a vehicle is allowed to reach customer i before the beginning of customer i's time window, e_i, but the vehicle cannot depart from customer i before e_i.

The two primary TSPTW objective functions considered in the literature are: (1) minimize the sum of the arc traversal costs along the tour and (2) minimize the time to return to the depot. We deal with the former objective function, $f(\Im) = \sum_{i=0}^{n} c(a_i)$, in order to make comparisons with the results of Wolfler Calvo (2000) and Gendreau et al. (1998). In order to check feasibility with respect to the time windows, we track the arrival time at i^{th} customer, A_{p_i}, and the time at which service starts at the i^{th} customer, D_{p_i} (which corresponds to the departure time from the i^{th} customer for the case of zero service time).

The TSPTW is composed of two main components, a traveling salesman problem and a scheduling problem. The TSP itself is an NP-hard optimization problem, and the scheduling aspect, with release dates and due dates, presents additional feasibility difficulties. Using a penalty method approach, we partially decompose these two components and conduct a heuristic search. We consider infeasible solutions by relaxing the time window constraints $\{D_{p_i} \leq l_{p_i} : i = 1, \ldots, n\}$ into the objective function with a penalty function of the form

$$p(\Im) \quad = \quad \sum_{i=1}^{n} \left[\max\left\{0, D_{p_i} - l_{p_i}\right\}\right]^s, \tag{1}$$

for some $s > 0$. Hadj-Alouane and Bean (1997) prove that for a sufficiently large nonnegative penalty multiplier, λ, penalty functions as defined in (1) maintain strong duality between the relaxation and the original formulation. We express our relaxed version of the TSPTW as

$\mathbf{RP}(\lambda)$ minimize $v(\Im, \lambda) = \sum_{i=0}^{n} c(a_i) + \lambda \sum_{i=1}^{n} \max\{0, D_{p_i} - l_{p_i}\}$

subject to:

$A_{p_i} = D_{p_{i-1}} + c(a_{i-1})$ for $i = 1, \ldots, n+1$;

$D_{p_i} = \max\{A_{p_i}, e_{p_i}\}$ for $i = 1, \ldots, n$;

$D_{p_0} = 0$,

$p_i \in \{1, 2, \ldots, n\}$ for $i = 1, \ldots n$;

$p_i \neq p_j$ for $i, j = 1, \ldots, n$, $i \neq j$;

$p_0 = 0$,

$p_{n+1} = 0$.

Note that if we were to consider the minimization of tour completion time, we could track the waiting time of the vehicle at each position of the tour, $W_{p_i} = D_{p_i} - A_{p_i}$, for $i = 1, \ldots, n$. The term $\sum_{i=1}^{n} W_{p_i}$ would then be added to the objective function in $\mathbf{RP}(\lambda)$.

3. Compressed Annealing Approach to the TSPTW

We focus on solving the TSPTW via an implementation of compressed annealing, a variant of simulated annealing, on the relaxation $\mathbf{RP}(\lambda)$. Simulated annealing is a stochastic local search method analogous to physical annealing, the process of melting and then slowly cooling a solid so that the substance reaches its lowest energy state. By accounting for the likelihood of a particular molecular configuration at a given temperature, Metropolis et al. (1953) develops a Monte Carlo simulation method for sampling molecular energy at a given temperature in the annealing process. Extending the Metropolis algorithm, Kirkpatrick et al. (1983) and Cerny (1985) independently introduce simulated annealing, a probabilistic search procedure for solving combinatorial optimization problems. Overviews of simulated annealing and its applications can be found in van Laarhoven and Aarts (1987) and Dowsland (1993).

Theodoracatos and Grimsley (1995) and Morse (1997) extend simulated annealing with an ad hoc introduction of a variable penalty multiplier (λ) to complement the traditional simulated annealing parameter called temperature (τ). Maintaining the physical analogy of annealing, we call the value of the penalty multiplier "pressure" and refer to the dual-parameterized annealing algorithm as "compressed" annealing. In the context of the TSPTW, temperature controls the probability of transition to a more costly route while pressure controls the probability of transition to an infeasible route with respect to the time windows. In the following subsections, we present a refined approach for simultaneously varying the pressure and temperature over an annealing run.

Table 1: Outline of Compressed Annealing

Initialize best tour found, $\Im_{best}$, so that $f(\Im_{best}) = \infty$ and $p(\Im_{best}) = 0$.
Generate initial tour, $\Im$.
Let $k = 0$.
Set initial temperature and pressure, τ_k and λ_k.
Set ι, the number of iterations at each temperature/pressure.
  Repeat:
    Let *counter* = 0.
    Repeat:
      Increment *counter* by 1.
      Randomly generate y, a neighbor tour of $\Im$.
      With probability $\exp\left(\frac{-(v(y,\lambda_k)-v(\Im,\lambda_k))^+}{\tau_k}\right)$, let $\Im = y$.
      If $p(\Im) \leq p(\Im_{best})$ and $f(\Im) < f(\Im_{best})$, let $\Im_{best} = \Im$.
    Until *counter* is equal to ι.
    Increment k by 1.
    Update τ_k and λ_k according to cooling and compression schedules.
  Until termination criterion satisfied.

3.1. Cooling and Compression

The search behavior of compressed annealing is directly affected by the manner in which the temperature and pressure parameters are respectively decreased and increased during the annealing run. Every ι iterations, the values of temperature and pressure are respectively updated according to a cooling schedule $\{\tau_0, \tau_1, \ldots\}$ and compression schedule $\{\lambda_0, \lambda_1, \ldots\}$. Note that τ_k and λ_k are the values of temperature and pressure from iteration $k\iota + 1$ to iteration $(k + 1)\iota$. Refer to Table 1 for an algorithmic outline of compressed annealing. In particular, notice that we update "best tour found" only when encountering a feasible solution with that improves upon the $f(\Im_{best})$. This reflects the objective of finding a minimum-cost tour that is feasible with respect to time windows. If a problem instance proves to be infeasible and we wish to identify a "good" infeasible solution, we can initialize $p(\Im_{best})$ appropriately and modify the update logic.

Ohlmann et al. (2004) discover that joint cooling and compression schedules should have decreasing derivatives to ensure convergence to the set of global minima. While the theoretical rates of cooling and compression are much too slow to be practical, they supply insight on appropriate "shapes" for simultaneous cooling and compression schedules (see Figure 1). Combining this intuition with observations documented in the literature, we implement a geometric cooling schedule (Dowsland, 1993) and a limited exponential compression schedule (Ohlmann et al., 2004). For parameters $0 \leq \beta \leq 1$, $\gamma \geq 0$, $\lambda_0 \geq 0$, $\hat{\lambda} \geq 0$, these

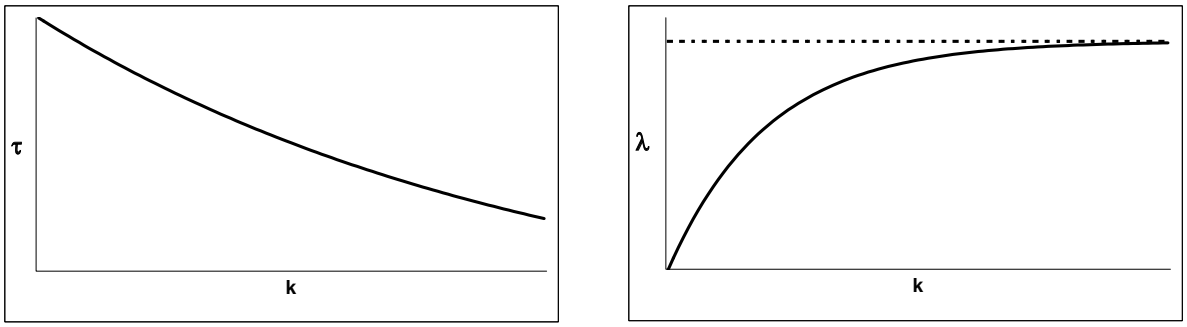

Figure 1: Demonstration of practical cooling and compression schedules.

schedules are formally defined by

$$\begin{aligned} \tau_{k+1} &= \beta\tau_k, \text{ and} \\ \lambda_{k+1} &= \hat{\lambda}\left(1 - \frac{(\hat{\lambda} - \lambda_0)}{\hat{\lambda}} e^{-\gamma k}\right). \end{aligned}$$

Values of the cooling parameter, β, typically range from 0.80 to 0.99 and values of the compression parameter, γ, usually vary from 0.01 to 0.1. To apply these schedules, initial values of temperature and pressure (τ_0 and λ_0) still need to be determined, as well as a maximum pressure ($\hat{\lambda}$). The limited exponential form of compression allows the convenience of simply setting $\lambda_0 = 0$, but more care must be taken in setting τ_0 and $\hat{\lambda}$.

The initial value of temperature must be selected so that early in the algorithm, the probability of accepting uphill transitions is close to 1; this allows the algorithm sufficient mobility to search the solution space. However, setting the temperature prohibitively high results in long computation times or poor convergence. Setting the initial temperature takes on increased importance in the presence of pressure, as setting τ_0 excessively high wastes the benefit of searching a "relaxed" topography in the sense that the search is random rather than guided by a tendency to go downhill.

As a preliminary step in parameter initialization, we generate $\mathcal{R}$, a set of $2r$ solutions obtained by randomly sampling r pairs of neighbor solutions. To generate these neighbor solutions, we utilize a 1-shift neighborhood as described in §3.4. In our testing, we find that setting $r = 5000$ provides sufficient information about the local topology at an acceptable computational cost. We utilize the information from $\mathcal{R}$ to specify an appropriate initial value of temperature by adapting techniques from van Laarhoven and Aarts (1987) and Dowsland (1993). First, we specify χ_0, the percentage of proposed uphill transitions that we require to be accepted at τ_0. Values of χ_0 typically range from 0.80 to 0.99. Computing $\overline{|\Delta v|}$, the average absolute difference in objective function over the n sample transitions composing $\mathcal{R}$, we determine the initial temperature as

$$\tau_0 = \frac{\overline{|\Delta v|}}{\ln\left(\frac{1}{\chi_0}\right)}.$$

At this value of initial temperature, the actual acceptance ratio over a trial loop of iterations of compressed annealing is monitored. If the actual acceptance ratio is less than χ_0, then τ_0 is reset at 1.5 times its current value and re-evaluated over a loop of iterations. This procedure is continued until the observed acceptance ratio for a loop of iterations equals or exceeds χ_0.

As shown by the theoretical analysis in Ohlmann et al. (2004), there exists a pressure cap, λ^*, beyond which further compression serves only to exaggerate the solution topography's features. Therefore, an ideal practical compression schedule would gradually increase λ from an initial value of zero to λ^*, allowing the algorithm to explore the solution space via solutions infeasible in terms of the relaxed constraints. Unfortunately, determining a tight upper bound on λ^* using only the limited information from the sample $\mathcal{R}$ is difficult. Nonetheless, we present an approach that, while not guaranteeing an upper bound on λ^*, can be experimentally calibrated to determine an approximation of the pressure cap.

To approximate the pressure cap, we introduce an additional parameter, $0 \leq \kappa \leq 1$ to determine our estimate, $\hat{\lambda}$. The value of κ represents the percentage of the objective function value that is composed of the penalty term when $\lambda = \hat{\lambda}$. Our pressure cap approximation is given by

$$\hat{\lambda} = \max_{\Im \in \mathcal{R}} \left\{ \frac{f(\Im)}{p(\Im)} \frac{\kappa}{1-\kappa} \right\}, \tag{2}$$

where values of κ ranging from 0.75 to $0.\overline{99}$ have demonstrated computational promise.

3.2. Iterations Per Temperature/Pressure Setting

Theoretical research on simulated annealing suggests that the system should be allowed to converge to its stationary distribution at each temperature setting. Unfortunately, the number of iterations necessary to approach the stationary distribution is exponential in problem size (Van Laarhoven and Aarts, 1987). In practice, the length of the Markov chain at each temperature is usually related to the size of the neighborhood structure or even the solution space. Bonomi and Lutton (1984) set the number of iterations at each temperature to a value depending polynomially on the size of the problem. An alternate approach determines the length of the k^{th} Markov chain by not allowing a temperature reduction until a minimum number of transitions have been accepted or a maximum number of iterations has been attained. In this manner, Kirkpatrick et al. (1983) let the length of the k^{th} Markov chain be dependent on k. Using problem size as an initial guideline, we fine-tune ι (the number of iterations per inner loop) through experimental testing to obtain an effective setting for a wide range of problem instances.

3.3. Termination Criterion

There have been numerous stopping conditions reported in the literature. Bonomi and Lutton (1984) fix the number of temperature values for which the algorithm is executed. Johnson et al. (1989) terminate the algorithm when the percentage of accepted moves drops below a threshold for a number of iterations. We

Table 2: Compressed Annealing Parameters for TSPTW

Parameter	Value
cooling coefficient (β)	.95
initial acceptance ratio (χ_0)	.94
compression coefficient (γ)	.06
pressure cap ratio (κ)	.9999
iterations per temperature (ι)	30000
minimum number of temperature changes	100

implement a hybrid of these two approaches by monitoring the mobility of the algorithm while also requiring a minimum number of iterations. Precisely, we terminate the compressed annealing runs when the best tour found has not been updated in the last 75 temperature/pressure changes while requiring a minimum of 100 total temperature changes.

3.4. Parameter Calibration

The need for potentially tedious parameter calibration is often a detractor in metaheuristic approaches. In our implementation, we avoid ad hoc parameter-tuning by utilizing the statistical design approach of Coy et al. (2000) to systematically determine robust parameter settings. The procedure outline by Coy et al. is performed in four steps. To begin, we select a collection of 15 individual data instances (varying in the number of customers and width of the time windows). We then run a pilot study to determine appropriate initial values for each parameter and a range over which good parameter settings are likely to be found. Using these initial values and ranges, we next use a 2^{m-1} fractional factorial experimental design, where m is the number of parameters, to run a series of experimental runs. With the results of the experimental runs for the 2^{6-1} different parameter settings, we use linear regression to find search directions for each of the parameter values. Finally, we run an additional set of computation experiments, each time updating the parameters in small increments of the search direction. We continue this process until the best solution remains the same over a number of iterations. For further details regarding the parameter-setting methodology, please consult Coy et al. (2000).

While the algorithm performs well over a relatively wide range of parameters, we suggest a robust set in Table 2. We also perform preliminary computational experiments to establish an effective neighborhood structure. We find that a 1-shift neighborhood scheme, in which a single customer and its new insertion position are randomly selected, results in quality solutions. Computational findings of Cheh et al. (1991) support this choice of neighborhood. Additionally, Carlton and Barnes (1996) implement a similar neighborhood structure for their application of tabu search on the TSPTW.

4. Computational Experience

We implement the compressed annealing algorithm in C++ and run the code on a Pentium 4 2.66 gigahertz processor with 1 gigabyte of RAM. We test the performance of the compressed annealing algorithm on five different sets of data (400 total instances) taken from the literature. The data sets are available via this journal's Online Supplements web page at http://joc.pubs.informs.org/OnlineSupplements.html. These sets are:

1. Thirty instances generated by Potvin and Bengio (1996) as individual route instances on Solomon's RC2 VRPTW instances (Solomon, 1987). Solomon's RC2 instances contain a mix of randomly-spaced and clustered customers.

2. Seventy instances proposed and solved to optimality by Langevin et al. (1993). The Langevin instances include problems of 20, 40, and 60 customers with time windows of 20, 40, and 60 time units. Due to their unavailability, we are unable to test the compressed annealing algorithm on the instances with 20 customers and time windows of 20 time units or the instances with 40 customers and time windows of 30 units.

3. One hundred thirty five instances proposed and solved to optimality by Dumas et al. (1995). The Dumas instances include problems considering between 20 and 200 customers with time window widths ranging from 20 to 100 time units.

4. One hundred forty instances proposed by Gendreau et al. (1998). The Gendreau instances consider the effect of widening time windows. A majority of the Gendreau instances are the same as the instances proposed by Dumas et al. (1995), except that the time windows have been systematically extended by 100 time units, resulting in time windows ranging from 120 to 200 time units in increments of 20.

5. Twenty five instances that have not previously appeared in the literature. We generate these instances by taking the 150 and 200 customer instances from Dumas et al. (1995) and systematically extending the time windows by 100 time units.

The results reported for the data sets from Dumas et al. (1995) and Gendreau et al. (1998) are averages over 5 instances for each class of problems (where a class of problems is distinguished by the number of customers and time-window width). Analogously, the results for the Langevin et al. (1993) sets are reported as averages over 10 instances for each class of problems.

Since compressed annealing is a stochastic search algorithm, we perform 10 runs from randomly generated starting solutions for each individual problem instance. While our results show that compressed annealing is generally robust with respect to its starting solution, it is unable to converge to a feasible solution for a few individual starting points generated in our testing. For these cases, we report (in parenthesis next to the reported average solution) the number of starting solutions that resulted in feasible TSPTW solutions.

Thus, in the absence of parenthesis, each table entry for compressed annealing's average solution on the Langevin sets is calculated over 100 total runs (10 runs on each of 10 instances). Similarly, for the Dumas and Gendreau sets, each table entry for the average compressed annealing results is calculated over 50 total runs (10 runs on each of 5 instances).

To accurately portray compressed annealing's performance with respect to the benchmarks discussed in Section 4.2, we present best-found solution values in addition to average solution values and solution quality variability. For each instance, we find a best solution value (the minimum value found over 10 runs). For the results reported by aggregating instances by problem class (Langevin et al., 1993; Dumas et al., 1995; Gendreau et al., 1998), the reported best solution value is the average of the best-found solution values for each individual instance in the class. We measure the variability of solution quality for the aggregated results by calculating a pooled estimate of the common variance for each of the instances in the problem class. The pooled estimate is given by $\sqrt{SS_E/(N-a)}$, where N is the total number of runs over the entire problem class, a is the number of different instances within the problem class, and SS_E is the sum of the squares due to error within instances. SS_E is computed according to $\sum_{i=1}^{a}\sum_{j=1}^{10}(y_{ij}-\bar{y}_{i\cdot})^2$ where y_{ij} denotes the solution value obtained by compressed annealing on run j of instance i and $\bar{y}_{i\cdot}$ denotes the average solution value obtained by compressed annealing on instance i (Montgomery, 2001).

Following the convention of van Laarhoven, Aarts, and Lenstra (1992), we report the average computation time (in CPU seconds) and the variability of computation time. We calculate these measures similarly to the average solution value and the variability of solution quality, respectively.

4.1. Search Advantage of Variable Penalty Multiplier

In this section, we present computational evidence to demonstrate that it is often difficult to determine a "good" value of a static penalty multiplier. A "large" penalty multiplier prevents convergence to infeasible solutions, but may retard the search of the solution space. On the other hand, fixing the penalty multiplier at a "small" value may be amenable to a less restricted neighborhood search, but may not satisfactorily differentiate between feasible and infeasible solutions.

Compressed annealing addresses the difficulty of selecting an appropriate multiplier value through its variable penalty approach. To quantify the benefits of varying pressure over the annealing run, we compare the performance of compressed annealing to the results obtained from implementing simulated annealing with a fixed penalty multiplier, $\bar{\lambda}$. To allow for a suitable comparison to compressed annealing, we test static simulated annealing with four different values of $\bar{\lambda}$. Using values of $\kappa = 0.25$, 0.50, 0.75, and 0.99 in (2), we calculate varying magnitudes of the fixed penalty multiplier, $\bar{\lambda}$, using a calculation similar to that used for $\hat{\lambda}$ in the compressed annealing approach. We then implement the four fixed-penalty annealing algorithms using the applicable parameters values found in Table 2.

In Table 3, we present the results of both compressed annealing and the static penalty versions of

annealing on the Solomon instances. For each instance, we report the average solution value for the five different approaches. We highlight (in bold-face type) the average solution value that attains the minimum value for each instance.

Table 3 validates the benefit of a variable penalty multiplier versus a static penalty multiplier. For relatively small fixed multiplier values ($\kappa = 0.25, 0.50$), simulated annealing rarely returns any feasible solutions for instances where $n > 20$. However, for instance *rc208(1)*, the best solution among the five approaches is obtained by using a fixed multiplier corresponding to $\kappa = 0.50$.

As we increase the fixed multiplier value ($\kappa = 0.75, 0.99$), we are more likely to converge to a feasible solution. While static annealing with $\kappa = 0.75$ obtains the sole best solution for *rc204(2)*, its performance is not robust and still fails to converges for several instances. The static annealing algorithm with $\kappa = 0.99$ returns feasible solutions for all instances, but its solutions are never better than those returned by compressed annealing. Furthermore, in cases which the fixed-penalty annealing with $\kappa = 0.25$, 0.50, or 0.75 return feasible solutions, one of them always returns a solution value superior to that obtained by the fixed annealing algorithm with $\kappa = 0.99$. It is important to note that the level of penalization returning the best fixed-penalty solution is dependent on the instance being solved.

In summary, the results in Table 3 suggest that, for a particular instance, we may be able to find a fixed penalty multiplier value that allows an effective local search. However, these static values are generally not robust; no one multiplier value performs particularly well for a variety of instances. In contrast, for compressed annealing, we can define a sequence of multiplier values that return good solutions across a wide variety of problem instances.

After testing the fixed penalty implementations of simulated annealing on other data sets from the literature, we find that Table 3's results on the Solomon sets are representative of the algorithm's shortcomings. In particular, for data sets with wide time windows, fixed penalty annealing returns feasible solutions only at high levels of penalization. This extreme penalization restricts the search and results in poor solutions.

4.2. Benchmarking

To evaluate the suitability of compressed annealing for the TSPTW, we compare the performance of compressed annealing to the performance of exact and heuristic solution methods reported in the literature. We reserve discussion of the relative computation times to the end of the section.

Table 4 compares the performance of compressed annealing to the previously best-known solutions for the Solomon sets, as obtained by the heuristic approaches of Wolfler Calvo (2000) and Gendreau et al. (1998). For each instance, we provide the best solution obtained in 10 compressed annealing runs as well as the average solution, the standard deviation of solution values, the average CPU time, and the standard deviation of CPU times. For the heuristic results of Wolfler Calvo (2000) and Gendreau et al. (1998), we report the solution value and CPU time for each instance. We denote current best-known solution values with bold-face

Table 3: Static versus Variable Penalty Multiplier Comparison for Individual Routes from VRPTW Instances in Solomon (1987)

Data Set		Compressed Annealing	SA: $\kappa = 0.25$	SA: $\kappa = 0.50$	SA: $\kappa = 0.75$	SA: $\kappa = 0.99$
		Avg. Solution	Avg. Solution	Avg. Solution	Avg. Solution	Avg. Solution
Problem	n	Value	Value	Value	Value	Value
rc201 (1)	19	**444.54**	-	**444.54** (5)	**444.54**	444.63
rc201 (2)	25	**711.54**	-	-	-	**711.54**
rc201 (3)	31	**790.61**	-	-	-	792.04
rc201 (4)	25	**793.64**	-	-	-	**793.64**
rc202 (1)	32	**771.99**	-	-		774.68
rc202 (2)	13	**304.14**	304.45	305.23	304.95 (8)	306.49
rc202 (3)	28	**837.72**	-	-	-	838.02
rc202 (4)	27	**793.03**	-	-	793.03	798.63
rc203 (1)	18	**453.48**	453.48	**453.48**	**453.48**	458.73
rc203 (2)	32	**784.16**	-	-	-	813.40
rc203 (3)	36	**817.53**	-	-	-	827.62
rc203 (4)	14	**314.29**	338.54	332.22	329.03	333.61
rc204 (1)	44	**880.37** (8)	-	-	-	889.25
rc204 (2)	32	667.76	-	-	**666.65**	726.62
rc204 (3)	33	**459.38**	518.69	515.26	515.10	554.34
rc205 (1)	13	**343.21**	**343.21**	**343.21**	**343.21**	**343.21**
rc205 (2)	26	**755.93**	-	-	-	756.20
rc205 (3)	34	**825.06**	-	-	-	**825.06**
rc205 (4)	27	**760.66**	-	-	-	761.74
rc206 (1)	3	**117.85**	**117.85**	**117.85**	**117.85**	**117.85**
rc206 (2)	36	**828.16**	-	-	-	834.32
rc206 (3)	24	**574.42**	-	-	-	582.91
rc206 (4)	37	**832.26**	-	-	-	837.33
rc207 (1)	33	**732.68**	-	-	**732.68** (3)	743.02
rc207 (2)	30	**701.25**	-	-	**701.25** (3)	709.85
rc207 (3)	32	**682.62**	-	-	-	699.84
rc207 (4)	5	**119.64**	**119.64**	**119.64**	**119.64**	**119.64**
rc208(1)	37	793.99	-	**791.53** (3)	793.99	833.43
rc208(2)	28	**533.78**	-	**533.78**	**533.78**	592.73
rc208(3)	35	**693.03**	-	-	*642.42 (1)*	729.68

font. Table 4 also indicates the percentage difference (Δ) between compressed annealing's best solution and the best-known solution calculated as (best known solution - compressed annealing solution)/(compressed annealing solution).

As the results show, compressed annealing finds a new best-known solution in 12 of the 30 instances and matches the previously best-known solution in 17 other instances. In addition, compressed annealing returns solutions equal to or better than the solutions returned by the insertion heuristic of Gendreau et al. (1998) on all instances. Furthermore, compressed annealing obtains feasible solutions in all 30 instances, while the assignment heuristic of Wolfler Calvo (2000) reports feasible solutions in only 28 instances. For the entire set of 30 instances, compressed annealing obtains the best-known solution in 29 instances, versus 18 and 12 best-known solutions for Wolfler Calvo (2000) and Gendreau et al. (1998) respectively. We also note that the standard deviation of the solution values is relatively low compared to the overall route times for each instance. This observation and the high-quality average solution values suggest that compressed annealing consistently converges to good solutions.

In Table 5, we present solution values for the TSPTW instances of Langevin et al. (1993). For each customer-time window class, we list the average solution value over the ten different instances. We compare compressed annealing to known optimal solutions and the solutions obtained by the heuristic procedure in Wolfler Calvo (2000). In the manner of Wolfler Calvo (2000), we calculate the percentage difference (Δ) as (compressed annealing solution - optimal solution)/(optimal solution). Compressed annealing exhibits promising behavior; it achieves the optimal solution on three of four instances with 20 and 40 customers and matches Wolfler Calvo's solutions on all instances. Direct comparison to optimal solutions is not possible for the problems with 60 customers because the optimal solution is only known in seven of the ten instances with 20-minute time windows, eight of the ten instances with 30-minute time windows, and seven of the ten instances with 40-minute time windows. In these 60-customer cases, compressed annealing matches the solutions obtained in Wolfler Calvo (2000). In addition, compressed annealing displays no variance in solution quality for five of the seven instances.

Table 6 provides computational results for the Dumas instances. For each customer-time window class, we list the average solution value over the five different instances. We report results from the exact solution method in Dumas et al. (1995), compressed annealing, and the best-known heuristic solution. Except where noted otherwise, the best-known heuristic solutions were found by Wolfler Calvo (2000). We also present the percentage difference (Δ) between the optimal solution and the best compressed annealing solution. The best compressed annealing solution matches the optimal solution in 25 of the 27 instances while outperforming the previously best-known heuristic solution in 10 cases and matching it on the other 17. Low solution variability and quality average solution values confirm compressed annealing's consistency.

As discussed in the literature review, known exact solution methods for the TSPTW are unable to find solutions to problems with wide time windows. Consequently, the wide time-window Gendreau instances are important in determining the value of a heuristic solution method. Table 7 presents a comparison of com-

Table 4: Results on Individual Routes from VRPTW instances in Solomon (1987)

Data Set		Compressed Annealing					Wolfler Calvo (2000)		Gendreau et al. (1998)		
Problem	n	Best Solution Value	Avg. Solution Value	Value Avg. σ	Avg. CPU Sec.	CPU Avg. σ	Solution Value	CPU Sec.	Solution Value	CPU Sec.	Δ
rc201 (1)	19	**444.54**	**444.54**	0.00	5.1	0.32	**444.54**	0	**444.54**	3.00	0.0%
rc201 (2)	25	**711.54**	**711.54**	0.00	5.8	0.42	**711.54**	0	712.91	6.98	0.0%
rc201 (3)	31	**790.61**	**790.61**	0.00	6.0	0.94	**790.61**	3	795.44	14.98	0.0%
rc201 (4)	25	**793.64**	**793.64**	0.00	4.5	0.71	**793.64**	0	**793.64**	6.00	0.0%
rc202 (1)	32	**771.78**	771.99	0.20	5.8	0.42	772.18	8	772.18	10.55	0.1%
rc202 (2)	13	**304.14**	**304.14**	0.00	4.6	0.70	**304.14**	0	**304.14**	2.35	0.0%
rc202 (3)	28	**837.72**	**837.72**	0.00	5.1	0.32	839.58	0	839.58	6.97	0.2%
rc202 (4)	27	**793.03**	**793.03**	0.00	4.9	0.99	**793.03**	2	**793.03**	11.55	0.0%
rc203 (1)	18	**453.48**	**453.48**	0.00	3.9	0.32	**453.48**	0	**453.48**	4.03	0.0%
rc203 (2)	32	**784.16**	**784.16**	0.00	6.1	0.32	**784.16**	4	**784.16**	15.67	0.0%
rc203 (3)	36	**817.53**	**817.53**	0.00	6.9	0.32	819.42	14	842.25	16.02	0.2%
rc203 (4)	14	**314.29**	**314.29**	0.00	3.4	0.52	**314.29**	0	**314.29**	2.98	0.0%
rc204 (1)	44	878.64	880.37 (8)	3.22	6.8	1.03	**868.76**	35	897.09	26.43	-1.1%
rc204 (2)	32	**662.16**	667.76	9.83	6.3	0.67	665.96	8	679.26	15.90	0.6%
rc204 (3)	33	**455.03**	459.38	2.29	4.5	0.53	**455.03**	4	460.24	11.18	0.0%
rc205 (1)	13	**343.21**	**343.21**	0.00	3.9	0.32	**343.21**	0	**343.21**	1.13	0.0%
rc205 (2)	26	**755.93**	**755.93**	0.00	6.3	1.34	**755.93**	0	**755.93**	7.33	0.0%
rc205 (3)	34	**825.06**	**825.06**	0.00	6.0	0.00	(825.06)	(21.00)	**825.06**	42.90	0.0%
rc205 (4)	27	**760.47**	760.66	0.61	4.6	0.97	-	-	762.41	6.58	0.3%
rc206 (1)	3	**117.85**	**117.85**	0.00	1.0	0.00	**117.85**	0	**117.85**	0.01	0.0%
rc206 (2)	36	**828.06**	829.57	4.44	6.2	0.42	842.17	10	842.17	33.47	1.7%
rc206 (3)	24	**574.42**	**574.42**	0.00	5.9	0.32	**574.42**	0	591.2	6.75	0.0%
rc206 (4)	37	**831.67**	832.26	1.85	7.0	0.00	837.54	8	845.04	31.48	0.7%
rc207 (1)	33	**732.68**	**732.68**	0.00	6.3	0.48	733.22	4	741.53	14.76	0.1%
rc207 (2)	30	**701.25**	**701.25**	0.00	7.0	0.00	-	-	718.09	16.28	2.4%
rc207 (3)	32	**682.40**	682.62	0.47	6.0	0.47	684.4	10	684.4	17.25	0.3%
rc207 (4)	5	**119.64**	**119.64**	0.00	2.0	0.00	**119.64**	0	**119.64**	0.01	0.0%
rc208(1)	37	**789.25**	793.99	2.86	6.4	0.70	**789.25**	10	799.19	26.58	0.0%
rc208(2)	28	**533.78**	**533.78**	0.00	5.5	0.53	537.33	2	543.41	20.53	0.7%
rc208(3)	35	**634.44**	639.03	5.90	6.7	0.82	649.11	8	660.15	25.63	2.3%

Table 5: Results on Instances Proposed by Langevin et al. (1993)

Data Set		Exact Algorithm		Compressed Annealing					Wolfler Calvo (2000)		
n	Time Window Width	Optimal Value	CPU Sec.	Best Solution Value	Avg. Solution Value	Value Avg. σ	Avg. CPU Sec.	CPU Avg. σ	Solution Value	CPU Sec.	Δ
20	30	724.7	0.4	**724.7**	**724.7**	0.0	2.4	0.8	**724.7**	0.0	0.0%
	40	721.5	0.7	**721.5**	**721.5**	0.0	3.4	0.6	**721.5**	0.0	0.0%
40	20	982.4	1.7	982.7	982.7	0.0	4.4	1.2	982.7	0.3	0.0%
	40	951.8	7.3	**951.8**	**951.8**	0.0	4.7	1.6	**951.8**	0.6	0.0%
60	20	-	-	**1215.7**	**1215.7**	0.0	5.6	2.4	**1215.7**	5.0	-
	30	-	-	**1183.2**	**1183.2**	0.3	8.1	2.7	**1183.2**	5.0	-
	40	-	-	**1160.8**	**1160.8**	0.8	9.0	2.1	**1160.8**	10.9	-

Table 6: Results on Instances Proposed by Dumas et al. (1995)

Data Set		Exact Algorithm		Compressed Annealing					Best-known Heuristic Values		
n	Time Window Width	Optimal Value	CPU Sec.	Best Solution Value	Avg. Solution Value	Value Avg. σ	Avg. CPU Sec.	CPU Avg. σ	Solution Value	CPU Sec.	Δ
20	20	361.2	0.0	**361.2**	**361.2**	0.0	2.0	0.0	**361.2**	0.0	0.0%
	40	316.0	0.1	**316.0**	**316.0**	0.0	2.7	0.4	**316.0**	0.0	0.0%
	60	309.8	0.1	**309.8**	**309.8**	0.0	2.5	0.3	**309.8**	0.0	0.0%
	80	311.0	0.2	**311.0**	311.0 (49)	0.0	3.0	0.0	**311.0**	0.0	0.0%
	100	275.2	1.3	**275.2**	**275.2**	0.0	3.2	0.3	**275.2**	0.0	0.0%
40	20	486.6	0.1	**486.6**	**486.6**	0.0	3.8	0.4	**486.6**	3.0	0.0%
	40	461.0	0.0	**461.0**	**461.0**	0.0	5.1	0.5	**461.0**	3.0	0.0%
	60	416.4	4.4	**416.4**	416.5	0.2	6.0	0.5	**416.4**	4.8	0.0%
	80	399.8	7.5	**399.8**	399.8 (49)	0.0	6.2	0.2	**399.8**	5.2	0.0%
	100	377.0	31.4	**377.0**	377.5	1.2	6.6	0.4	**377.0**	5.6	0.0%
60	20	581.6	0.2	**581.6**	**581.6**	0.0	7.2	0.9	**581.6**	8.4	0.0%
	40	590.2	0.9	**590.2**	590.7 (47)	2.0	8.2	0.4	**590.2**[a]	36.8	0.0%
	60	560.0	6.8	**560.0**	560.0	0.2	8.5	0.4	**560.0**	20.2	0.0%
	80	508.0	46.6	**508.0**	509.3 (49)	1.6	8.6	0.3	509.0	18.0	0.0%
	100	514.8	199.8	**514.8**	516.5	3.0	8.8	0.4	516.4	26.2	0.0%
80	20	676.6	0.4	**676.6**	**676.6**	0.0	11.3	0.4	**676.6**	43.4	0.0%
	40	630.0	2.7	**630.0**	630.2	0.6	11.5	0.4	**630.0**	69.2	0.0%
	60	606.4	55.3	**606.4**	607.0	1.4	12.0	0.3	609.2[b]	4.0	0.0%
	80	593.8	220.3	**593.8**	594.1 (48)	2.3	11.5	0.5	594.4	59.6	0.0%
100	20	757.6	0.6	**757.6**	757.8	0.8	15.4	0.4	**757.6**[a]	175.0	0.0%
	40	701.8	7.4	**701.8**	702.4 (46)	1.1	15.7	0.3	702.8[b]	3.0	0.0%
	60	696.6	108.0	**696.6**	697.2 (48)	1.2	15.9	0.5	**696.6**	148.0	0.0%
150	20	868.4	2.4	**868.4**	869.2 (49)	1.3	24.7	0.7	868.6	419.8	0.0%
	40	834.8	115.9	**834.8**	836.2	2.7	25.2	0.7	836.6[b]	9.0	0.0%
	60	805.0	463.0	818.8	820.2 (44)	4.8	25.6	0.7	820.4	630.0	1.7%
200	20	1009.0	6.7	**1009.0**	1010.0	2.1	35.1	1.7	1010.0	1456.2	0.0%
	40	984.2	251.4	984.6	986.1 (45)	3.4	35.2	1.3	985.4	2105.8	0.0%

[a]The best-known heuristic solution value is found by the algorithm proposed by Gendreau et al. (1998)
[b]The best-known heuristic solution value is obtained by Carlton and Barnes (1996)

pressed annealing and the algorithms of Wolfler Calvo (2000) and Gendreau et al. (1998) on these Gendreau instances. We express the percentage difference between the best compressed annealing solution and the best-known heuristic solution value as (best heuristic solution - compressed annealing solution)/(compressed annealing solution). Compressed annealing obtains the best-known results on 20 of the 28 different sets of instances. It is also important to recognize that compressed annealing exhibits very little standard deviation among its solutions, suggesting that the algorithm consistently handles the wide time windows.

Finally, in Table 8, we present the results of extended time window instances generated from 150 and 200 customer instances of Dumas et al. (1995). We believe that these instances are an important contribution to the TSPTW benchmark sets because they show a solution method's ability to not only cope with wide time windows, but also with the large numbers of customers which are often encountered in industrial applications. While we cannot provide a comparison of solution values with other solution methods, our results do show that compressed annealing's solutions exhibit relatively low variability. This small variation is indicative of compressed annealing's ability to consistently handle both the wide time windows and the increased number of customers.

Because of differences in processor speed, memory, bus speed, and language implementation, run-time comparisons are difficult. However, processor comparisons suggest that our algorithm is slower than those of Wolfler Calvo (2000) and Gendreau et al. (1998). However, given today's processor speeds and the general nature of our implementation, our algorithm is certainly capable of solving reasonably large problems in adequate time. In addition, as the run-times in Tables 7 and 8 show, computation time for compressed annealing is only minimally affected by increasing numbers of customers and time-window widths. This result is in contrast to the performance of Wolfler Calvo's and Gendreau's algorithms under the same conditions. Thus, the result suggests that compressed annealing is particularly valuable in circumstances involving large numbers of customers or wide time windows. In addition, our computational experience has shown that, by revising the termination criteria such that the algorithm terminates when the best tour found has not been updated in 25 temperature/pressure changes, run times can be reduced by 20% to 30% for almost all problems. This reduction in computation time also only minimally reduces the quality of the average solution as most average solutions are still within 1% of the optimal or previously best-known heuristic solution.

5. Conclusions and Future Considerations

We have presented a solution approach to the TSPTW, a difficult combinatorial problem, utilizing compressed annealing. Using a variable penalty function and stochastic search, we consider solutions infeasible with respect to time windows during our search for optimal or near-optimal solutions. Computational testing on five series of TSPTW problems demonstrates the potential of the compressed annealing algorithm. Near-optimal solutions can be obtained at a reasonable computational cost in most cases, and feasible solutions are found in every instance. Compressed annealing compares favorably with benchmarks in the literature,

Table 7: Results on Instances Proposed by Gendreau et al. (1998)

Data Set		Compressed Annealing					Wolfler Calvo (2000)		Gendreau et al. (1998)		
n	Time Window Width	Best Solution Value	Avg. Solution Value	Value Avg. σ	Avg. CPU Sec.	CPU Avg. σ	Solution Value	CPU Sec.	Solution Value	CPU Sec.	Δ
20	120	**265.6**	**265.6**	0.0	3.1	0.4	267.2	0.0	269.2	4.1	1%
	140	**232.8**	**232.8**	0.0	3.9	0.3	259.6	0.0	263.8	4.4	12%
	160	**218.2**	**218.2**	0.0	4.0	0.1	260.0	0.0	261.2	4.8	19%
	180	**236.6**	**236.6**	0.0	4.0	0.1	244.6	0.0	259.8	6.0	3%
	200	**241.0**	**241.0**	0.0	4.1	0.2	243.0	0.4	245.2	6.3	1%
40	120	377.8	378.1	1.1	6.0	0.2	**360.0**	4.8	372.8	18.4	-5%
	140	364.4	364.7	1.6	6.0	0.1	**348.4**	9.4	356.2	18.9	-4%
	160	**326.8**	327.1	0.6	6.0	0.2	337.2	10.2	348.0	20.0	3%
	180	332.0	333.9	2.3	6.2	0.4	**326.8**	12.4	328.2	17.0	-2%
	200	**313.8**	315.0	1.0	6.3	0.4	315.2	16.2	326.2	22.8	0%
60	120	**451.0**	452.9	2.8	8.3	0.2	483.4	29.8	492.0	51.6	7%
	140	**452.4**	454.0 (48)	2.1	8.6	0.4	454.4	28.0	454.8	49.5	0%
	160	464.6	465.4	2.3	8.4	0.4	**448.6**	33.8	451.6	47.5	-3%
	180	**421.6**	425.2	4.4	8.6	0.4	432.8	40.6	439.2	52.3	3%
	200	**427.4**	430.8	5.0	8.4	0.3	428.0	57.0	439.6	43.5	0%
80	100	**579.2**	581.6	2.4	11.5	0.4	580.2	72.8	584.2	99.5	0%
	120	**541.4**	544.0	2.1	11.5	0.4	549.8	64.0	581.8	121.0	2%
	140	**509.8**	513.6	4.7	11.3	0.4	525.6	75.2	555.2	94.2	3%
	160	505.4	511.7	5.2	11.2	0.4	**502.8**	82.2	524.8	85.7	-1%
	180	502.0	505.9	4.0	11.4	0.3	**489.0**	116.2	511.0	99.0	-3%
	200	**481.8**	486.4	4.0	11.1	0.3	484.0	158.2	508.6	112.3	0%
100	80	**666.4**	668.1	2.6	15.9	0.4	668.0	139.2	675.6	118.1	0%
	100	**642.2**	645.0	2.6	14.6	0.5	644.0	118.6	671.2	129.5	0%
	120	**601.2**	603.7	2.1	15.0	0.4	614.4	167.5	624.6	204.2	2%
	140	**579.2**	582.5	3.2	14.9	0.4	591.4	200.6	634.6	207.7	2%
	160	584.0	588.8	3.8	15.0	0.6	**570.4**	214.2	585.2	215.6	-2%
	180	**561.6**	566.9	4.6	14.9	0.4	566.0	244.6	585.2	225.1	1%
	200	**555.4**	562.3	5.8	14.9	0.4	555.6	242.0	588.6	168.2	0%

Table 8: Results on Extensions of Dumas' 150 and 200 Customer Instances

Data Set		Compressed Annealing				
n	Time Window Width	Best Solution Value	Avg. Solution Value	Value Avg. σ	Avg. CPU Sec.	CPU Avg. σ
150	120	725.0	731.1	5.5	24.8	0.9
	140	697.6	705.4	6.7	24.9	0.7
	160	673.6	680.9	5.9	25.0	1.0
200	120	806.8	817.0	7.0	34.4	1.3
	140	804.6	812.6	6.7	35.2	1.0

obtaining best-known results in numerous instances.

The variable penalty approach of compressed annealing generally outperforms simulated annealing with a suitable static penalty method. For a traditional simulated annealing approach, setting a static penalty multiplier that allows an adequate search of the solution space often proves to be a difficult chore. The parameterized penalty multiplier within compressed annealing creates a dynamic search procedure resulting in good solutions to constrained combinatorial problems.

Future research may include further analysis of the effect of penalty functions on heuristic search. In the current implementation, we use a single penalty term to penalize time window violations for the n customers. A natural extension would involve the analysis of an annealing approach that relaxes multiple constraint types with distinct penalty terms.

Acknowledgments

Barrett Thomas would like to thank the University of Iowa's Old Gold Foundation for their partial support of this research. Jeffrey Ohlmann extends his appreciation to his advisors, James Bean and Shane Henderson, who made valuable comments in the formative stages of this work. The authors acknowledge Robert Hansen's assistance in improving the computer implementation of the algorithm. The authors also thank three anonymous referees for their useful comments. Additionally, the authors would like to thank Roberto Wolfler Calvo, Michel Gendreau, and Mihnea Stan for providing the data sets used in the computational testing.

References

Baker, E. 1983. An exact algorithm for the time constrained traveling salesman problem. *Oper. Res.* **31** 938–945.

Bonomi, E., J. Lutton. 1984. The N-city traveling salesman problem: statistical mechanics methods and Metropolis algorithm. *SIAM Rev.* **36** 551–568.

Carlton, W. B., J. W. Barnes. 1996. Solving the traveling-salesman problem with time windows using tabu search. *IIE Trans.* **28** 617–629.

Cerny, V. 1985. Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. *J. of Optim. Theory and Appl.* **45** 41–51.

Cheh, K., J. Goldberg, R. Askin. 1991. A note on the effect of neighborhood structure in simulated annealing. *Computers & Oper. Res.* **18** 537–547.

Christofides, N., A. Mingozzi, P. Toth. 1981. State space relaxation procedures for the computation of bounds to routing problems. *Networks* **11** 145–164.

Coy, S. P., B. L. Golden, G. C. Runger, E. A. Wasil. 2000. Using experimental design to find effective parameter settings for heuristics. *J. of Heuristics* **7** 77–97.

Dowsland, K. A. 1993. Simulated annealing. C. R. Reeves, ed. *Modern Heuristic Techniques for Combinatorial Problems.* John Wiley and Sons, New York. 20–69.

Dumas, Y., J. Desrosiers, E. Gélinas, M. M. Solomon. 1995. An optimal algorithm for the traveling salesman problem with time windows. *Oper. Res.* **43** 367–371.

Focacci, F., A. Lodi, M. Milano. 2002. A hybrid exact algorithm for the TSPTW. *INFORMS J. on Computing* **14** 403–417.

Gendreau, M., A. Hertz, G. Laporte. 1992. New insertion and postoptimization procedures for the traveling salesman problem. *Oper. Res.* **40** 1086–1094.

Gendreau, M., A. Hertz, G. Laporte, M. Stan. 1998. A generalized insertion heuristic for the traveling salesman problem with time windows. *Oper. Res.* **46** 330–335.

Hadj-Alouane, A., J. Bean. 1997. A genetic algorithm for the multiple-choice integer program. *Oper. Res.* **45** 92-101.

Johnson, D., C. Aragon, L. McGeoch, C. Schevon. 1989. Optimization by simulated annealing: an experimental evaluation; part I, graph partitioning. *Oper. Res.* **37** 865–892.

Kirkpatrick, S., C. Gellat, M. Vecchi. 1983. Optimization by simulated annealing. *Science* **220** 671–680.

Langevin, A., M. Desrochers, J. Desrosiers, S. Gélinas, F. Soumis. 1993. A two-commodity flow formulation for the traveling salesman and makespan problems with time windows. *Networks* **23** 631–640.

Metropolis, N., A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller. 1953. Equation of state calculations by fast computing machines. *J. of Chemical Physics* **21** 1087–1092.

Montgomery, D. C. 2001. *Design and Analysis of Experiments*. John Wiley & Sons, New York.

Morse, C. 1997. *Stochastic Equipment Replacement with Budget Constraints*. Ph.D. thesis, Industrial and Operations Engineering, University of Michigan, Ann Arbor, Michigan.

Ohlmann, J., J. Bean, S. Henderson. 2004. Convergence in probability of compressed annealing. *Math. of Oper. Res.* **29** 837–860.

Pesant, G., M. Gendreau, J.-Y. Potvin, J.-M. Rousseau. 1998. An exact constraint logic programming algorithm for the traveling salesman problem with time windows. *Transportation Sci.* **32** 12–29.

Pesant, G., M. Gendreau, J.-Y. Potvin, J.-M. Rousseau. 1999. On the flexibility of constraint programming models: from single to multiple time windows for the traveling salesman problem. *Eur. J. of Oper. Res.* **117** 253–263.

Potvin, J.-Y., S. Bengio. 1996. The vehicle routing problem with time windows part II: genetic search. *INFORMS J. on Computing* **8** 165–172.

Savelsbergh, M. W. P. 1985. Local search in routing problems with time windows. *Annals of Oper. Res.* **4** 285–305.

Solomon, M. M. 1987. Algorithms for the vehicle routing and scheduling problems with time windows. *Oper. Res.* **35** 254–265.

Theodoracatos, V., J. Grimsley. 1995. The optimal packing of arbitrarily-shaped polygons using simulated annealing and polynomial-time cooling schedules. *Comp. Methods in Appl. Mech. and Engrg.* **125** 53–70.

Van Laarhoven, P. J. M., E. H. L. Aarts. 1987. *Simulated Annealing*. P. Reidel Publishing Co., Dordecht, Netherlands.

Van Laarhoven, P. J. M., E. H. L. Aarts, J. K. Lenstra. 1992. Job shop scheduling by simulated annealing. *Oper. Res.* **40** 113–125.

Wolfler Calvo, R. 2000. A new heuristic for the traveling salesman problem with time windows. *Transportation Sci.* **34** 113–124.