Simultaneous Generalized Hill Climbing Algorithms for Addressing Sets of Discrete Optimization Problems

Diane E. Vaughan

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Industrial and Systems Engineering

Sheldon H. Jacobson, Co-Chair
C. Patrick Koelling, Co-Chair
Robert Rogers
Joel A. Nachlas
Ebru Bish

July 31, 2000
Blacksburg, Virginia

Keywords: Local Search, Generalized Hill Climbing Algorithms, Simulated Annealing, Markov Chains, Ergodicity, Traveling Salesman Problem, Manufacturing Applications

Simultaneous Generalized Hill Climbing Algorithms for Addressing Sets of Discrete Optimization Problems

Diane E. Vaughan

(ABSTRACT)

Generalized hill climbing (GHC) algorithms provide a framework for using local search algorithms to address intractable discrete optimization problems. Many well-known local search algorithms can be formulated as GHC algorithms, including simulated annealing, threshold accepting, Monte Carlo search, and pure local search (among others).

This dissertation develops a mathematical framework for simultaneously addressing a set of related discrete optimization problems using GHC algorithms. The resulting algorithms, termed s*imultaneous generalized hill climbing* (SGHC) algorithms, can be applied to a wide variety of sets of related discrete optimization problems. The SGHC algorithm probabilistically moves between these discrete optimization problems according to a problem generation probability function. This dissertation establishes that the problem generation probability function is a stochastic process that satisfies the Markov property. Therefore, given a SGHC algorithm, movement between these discrete optimization problems can be modeled as a Markov chain. Sufficient conditions that guarantee that this Markov chain has a uniform stationary probability distribution are presented. Moreover, sufficient conditions are obtained that guarantee that a SGHC algorithm will visit the globally optimal solution over all the problems in a set of related discrete optimization problems.

Computational results are presented with SGHC algorithms for a set of traveling salesman problems. For comparison purposes, GHC algorithms are also applied individually to each traveling salesman problem. These computational results suggest that optimal/near optimal solutions can often be reached more quickly using a SGHC algorithm.

Acknowledgements

I wish to first acknowledge my advisors Dr. Sheldon H. Jacobson and Dr. C. Patrick Koelling, their expertise, guidance, time and support made this dissertation possible. I especially wish to thank Dr. Sheldon H. Jacobson who has patiently assisted in my development as a researcher. I also wish to acknowledge and thank my other committee members Dr. Robert Rogers, Dr. Joel A. Nachlas and Dr. Ebru Bish for their time and support.

I would like to thank the Mathematics Department and the Industrial and Systems Engineering Department for their support of and dedication to graduate students. In particular, I thank Ms. Eileen Shugart and Ms. Lovedia Cole for their patience, kindness and commitment to their jobs.

I would like to thank Dr. Neal D. Glassman of the Air Force Office of Scientific Research for supporting the research contained in this dissertation (through AFSOR grants F49620-98-1-0111 and F49620-98-1-0432). I also would like to thank Dr. W. Garth Frazier of the Materials Process Design Branch of the Air Force Research Laboratory, Wright Patterson Air Force Base, and Mr. Enrique Medina of Austral Engineering and Software, Inc., of Athens, Ohio, for providing the initial motivation and an interesting military manufacturing problem application of the results developed in this dissertation. I also would like to thank Dr. Richard Nance, Director of the Systems Research Laboratory for his support of the work in this dissertation.

I would like to acknowledge my fellow students who have influenced my research directions, helped me develop my presentation skills and facilitated in developing and maintaining my interest in operations research and mathematics. In particular, I would like to thank Dr. Jeanne Atwell, Chris Orum, Lieutenant Colonel Darrall Henderson, Amy Anderson, Wendy Hageman Smith, Becker Sidney Smith, Derek E. Armstrong, and Elise Caruso.

I also would like to thank Dr. John E. Kobza, an excellent teacher and researcher, and Derek E. Armstrong and Dr. Sheldon Jacobson, my co-authors on the paper that inspired the SGHC algorithm. Lastly, I wish to thank the "GHC" team: Dr. Sheldon H. Jacobson, Lieutenant Colonel Darrall Henderson, Tevfik Aytemiz, Lieutenant Colonel (Dr.) Alan Johnson, Dr. Kelly Sullivan and Derek E. Armstrong, for making work a fun and intellectual experience.

Dedication

This dissertation is dedicated to several important people in my life who have made this effort possible. First and most importantly, this dissertation is dedicated to my husband Mark and our son Andrew, who cooked, cleaned and smiled to help with its completion and who fill our home with happiness. In addition, this dissertation is dedicated to Denise and Patricia, for their love and support.

This dissertation is also dedicated to Jim and Jeanne Atwell who were always there when we needed them. I would also like to dedicate this dissertation to Richard Hardy, whose advice about life and career decisions I took seriously.

Finally, I wish to dedicate this dissertation to my parents. I would like to thank the Makelas for lovingly accepting me, my dreams and my ambitions. I am forever indebted to the Vaughans, who have so many attributes that I admire and respect. I strive to be a reflection of them.

Contents

List of Figures

List of Tables

Chapter 1:

Introduction and Motivation

1.1 Simultaneous Generalized Hill Climbing Algorithms

Generalized hill climbing (GHC) algorithms (Jacobson et al. 1998) provide a framework for using local search algorithms to address discrete optimization problems. GHC algorithms include many local search algorithms, including simulated annealing (Fleischer 1995), threshold accepting (Dueck and Scheuer 1990), Monte Carlo search, and pure local search (Tovey 1983). The GHC algorithm framework allows for the development of convergence and performance properties that apply to *families* of GHC algorithms. Therefore, the GHC algorithm framework eliminates the need to investigate local search algorithms individually. For example, sufficient convergence conditions for a particular family of GHC algorithms that includes simulated annealing are presented in Johnson and Jacobson (2000a). This dissertation develops and studies a mathematical framework for computationally approaching several discrete optimization problems simultaneously using GHC algorithms. The resulting algorithms are termed *simultaneous generalized hill climbing* (SGHC) algorithms.

It is common to encounter several discrete optimization problems where a relationship between the solution spaces of the individual problems exists. In general, these problems are

approached individually. However, because of their similarities, the same computational tools can be effectively used to address them. For example, the Material Process Design Branch of the Air Force Research Laboratory, Wright Patterson Air Force Base (Dayton, Ohio, USA) is studying several similar discrete manufacturing process design optimization problems.

Discrete manufacturing process design optimization can be difficult due to the large number of design sequences and associated input parameter setting combinations that exist. GHC algorithms have been introduced to address such manufacturing design problems (Jacobson et al. 1998). Initial results with GHC algorithms required the manufacturing process design sequence to be fixed with the GHC algorithm used to identify optimal input parameter settings (Jacobson et al. 1998).

To motivate the development of SGHC algorithms, this dissertation introduces a new neighborhood function that allows GHC algorithms to be used to also identify the optimal discrete manufacturing process design sequence among a set of valid design sequences (Vaughan et al. 2000). Hence, this neighborhood function allows the GHC algorithm to simultaneously optimize over both the design sequences and the input parameters. Computational results are reported with an integrated blade rotor discrete manufacturing process design problem under study at the Materials Process Design Branch of the Air Force Research Laboratory.

This dissertation formally defines a class of sets of discrete optimization problems where a relationship similar to the one described for the manufacturing problem exists. A set of discrete optimization problems that is contained in this class is a set of *fundamentally related* discrete optimization problems. SGHC algorithms are designed to address sets of fundamentally related discrete optimization problems using GHC algorithms.

1.2 Research Goals

The objective of this dissertation is to introduce and study a general mathematical framework for simultaneously addressing a set of fundamentally related discrete optimization problems. A wide variety of manufacturing and service industry problems can be modeled as several discrete optimization problems that are typically addressed individually using local search algorithms. SGHC algorithms are a new approach that allows practitioners to make a single algorithm run over a set of fundamentally related discrete optimization problems.

This dissertation formally defines the class of fundamentally related sets of discrete optimization problems and develops a metric between elements in a set of fundamentally related discrete optimization problems (Vaughan et al. 2000). This metric is a tool for evaluating if it is advantageous to address a particular set of discrete optimization problems with a SGHC algorithm. Computational results for a set of manufacturing problems and a set of traveling salesman problems are presented that validate the usefulness of simultaneously addressing a set of discrete optimization problems using GHC.

The SGHC algorithm probabilistically moves between discrete optimization problems according to a problem generation probability function. The problem generation probability function is shown to be a stochastic process that satisfies the Markov property. Therefore, for a SGHC algorithm, movement between discrete optimization problems can be modeled as a Markov chain. Sufficient conditions that guarantee that this Markov chain has a uniform stationary probability distribution are provided. Additionally, sufficient conditions are presented that guarantee that a SGHC algorithm will visit the globally optimal solution over all the problems in a set of discrete optimization problems.

1.3 Research Questions

This dissertation investigates the following research questions

1. Can a mathematical framework be developed for simultaneously approaching several related discrete optimization problems?

2. Is there a metric that can be used to develop neighborhood functions for simultaneously approaching a set of discrete optimization problems? If so, can this metric be used to assess when it is possible to address several related discrete optimization problems simultaneously using GHC algorithms?

3. Given a set of related discrete optimization problems and a GHC algorithm for which convergence properties are known, can these properties be extended to guarantee convergence of a SGHC algorithm?

Chapter 2:

Literature Review

2.1 Discrete Optimization Problems

The study of discrete optimization problems is of growing interest and importance to industry since many real-world problems can be modeled as discrete optimization problems and due to advances in computational power (Hillier and Lieberman 1995). Discrete optimization problems are defined by a finite set of solutions, (i.e., *the solution space*), $\Omega=\{\omega_1, \omega_2, \ldots, \omega_{|\Omega|}\}$, together with an *objective function* $f:\Omega \rightarrow R$ (Garey and Johnson 1979). The objective function is a quantitative measure of the quality of each solution that assigns a real value to each element in the solution space (i.e., $f(\omega)\in R$, for all $\omega\in\Omega$) (Aarts and Lenstra 1997). The goal when addressing a discrete optimization problem is to find solutions that minimize/maximize the objective function. Without loss of generality, assume that all discrete optimization problems are minimization problems. A solution, $\omega^*\in\Omega$, that minimizes the objective function (i.e., $f(\omega^*)\leq f(\omega)$, for all $\omega\in\Omega$) is a *globally optimal solution*.

Discrete optimization problems can be defined by two complexity classes, *easy* (i.e., a globally optimal solution can be found in polynomial time in the size of the problem instance) and *hard* (i.e., in the class NP-hard). Garey and Johnson (1979) present a

discussion on the complexity of discrete optimization problems that can be classified as easy or hard. To address hard discrete optimization problems, local search algorithms are formulated with the goal of identifying good or near-optimal solutions (see Garey and Johnson 1979).

2.2 Local Search Algorithms

The use of local search algorithms can be traced back to the late 1950's and early 1960's (Aarts and Lenstra 1997). For example, (Bock 1958a,b, Croes 1958, Lin 1965, Reiter and Sherman 1965) introduced the first edge neighborhood structures for the traveling salesman problem (TSP). Today, there are many well-known local search algorithms for approaching discrete optimization problems including simulated annealing (Eglese 1990, Fleischer 1995), genetic algorithms (Liepins and Hilliard 1989), pure local search, threshold accepting (Dueck and Scheuer 1990), and tabu search strategies (Glover and Laguna 1997). All of these local search algorithms are designed with the goal of traversing the solution space in search of optimal/near optimal solutions.

2.2.1 Neighborhood Functions

To apply a local search algorithm to a discrete optimization problem a *neighborhood function*, $\eta:\Omega \rightarrow 2^{\Omega}$, where $\eta(\omega)\subset\Omega$ for all $\omega\in\Omega$, is required. Neighborhood functions allow the solution space to be traversed or searched by moving between solutions, hence they provide connections between solutions in the solution space. Generally, the choice of a neighborhood function is considered independent of the local search algorithm. However, a neighborhood function must be chosen such that all the solutions in the solution space (with neighborhood function η) are *reachable;* that is, for all $\omega',\omega''\in\Omega$, there exists a set of solutions $\omega_1, \omega_2, \ldots, \omega_m\in\Omega$ such that $\omega_r\in\eta(\omega_{r-1})$, r=1, 2, ..., m+1, where $\omega'\equiv\omega_0$ and $\omega''\equiv\omega_{m+1}$, ensuring that the solution space is not *fragmented*.

For a neighborhood function, η, a *locally optimal solution*, $\omega^L \in \Omega$, minimizes the objective function in its neighborhood (i.e., $f(\omega^L) \leq f(\omega)$, for all ω, $\omega \in \eta(\omega^L)$). Note that all globally optimal solutions are locally optimal solutions, however the converse is not necessary true. Jacobson and Yucesan (2000) use the objective function, f, and the neighborhood function, η, to decomposed solution space, Ω, into three mutually exclusive and exhaustive sets:

- a set of *G-local optima*, G (i.e., global optima),
- a set of *L-local optima*, $L \equiv L(\eta)$ (i.e., local optima that are not G-local optima),
- a set of *hill solutions*, H

Therefore, $G \cup L$ is the set of locally optimal solutions in Ω (associated with neighborhood structure η). By definition, $\Omega = G \cup L \cup H$, $G \cap L = \varnothing$, $G \cap H = \varnothing$, and $L \cap H = \varnothing$. Note that, also by definition, for all $\omega \in G$, $\eta(\omega) \cap L = \varnothing$, and for all $\omega \in L$, $\eta(\omega) \cap G = \varnothing$. Therefore, a G-local optimum and a L-local optimum cannot be neighbors of each other (Jacobson and Yucesan 2000). Figure 2.1 depicts a solution space, where solutions that are neighbors are connected with lines.

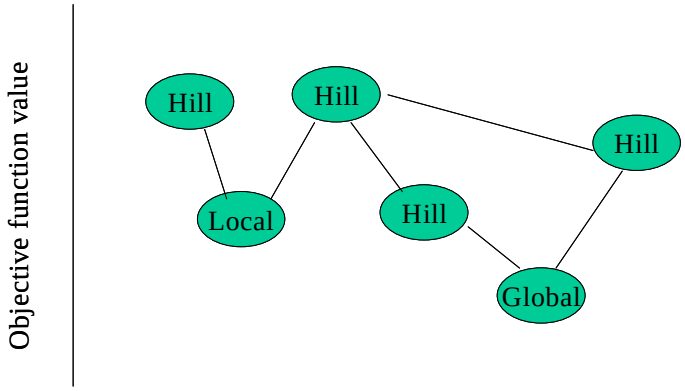

Figure 2.1: Locals, Hills and Globals

Many local search algorithms attempt to overcome the trappings of local optima. Simulated annealing is one such local search algorithm.

2.3 Simulated Annealing

Simulated annealing was introduced in the 1980's, independently by Kirkpatrick et al. (1982, 1983) and by Cerny (1985) (Aarts and Korst 1989). Simulated annealing mimics the annealing process for crystalline solids, where a solid is slowly cooled from an elevated temperature, with the objective of relaxing towards a low-energy state.

The literature contains several results on the asymptotic performance of simulated annealing algorithms. For simulated annealing algorithms with an exponential acceptance probability function, Mitra et al. (1986) and Hajek (1988) present conditions for three convergence properties: asymptotic independence of the starting conditions, convergence in distribution of the solutions generated, and convergence to a global optimum (Johnson and Jacobson 2000b). Anily and Federgruen (1987) extend these results to simulated annealing algorithms with general acceptance probabilities. Anily and Federgruen (1987) develop necessary and sufficient conditions for convergence and provide conditions for the reachability of the set of global optima. Schuur (1997) provides a description of acceptance functions that ensure the convergence of the associated simulated annealing algorithm to a globally optimal solution. Sufficient convergence conditions for a large family of GHC algorithms that includes simulated annealing are presented in Johnson and Jacobson (2000a, 2000b).

2.4 The Generalized Hill Climbing Algorithm

The GHC algorithm framework provides a structure for using local search algorithms to address intractable discrete optimization problems. The GHC algorithm framework contains many local search algorithms that seek to find optimal solutions for discrete optimization problems by allowing the algorithm to visit inferior solutions enroute to an optimal/near optimal solution. Figure 2.2 depicts the GHC pseudo-code.

Figure 2.2: The GHC Pseudo-Code

Set the outer loop counter bound K and the inner loop counter bounds N(k), k=1, 2, …, K
Define a set of hill climbing (random) variables R_k: $\Omega \times \Omega \rightarrow \Re \cup \{-\infty,+\infty\}$, k=1, 2, …, K
Set the iteration indices i=0, k=n=1
Select an initial solution $\omega(0) \in \Omega$
Repeat while $k \leq K$
  Repeat while $n \leq N(k)$
    Generate a solution $\omega \in \eta(\omega(i))$
    Calculate $\delta(\omega(i),\omega) = f(\omega)-f(\omega(i))$
    If $\delta(\omega(i),\omega) \leq 0$, then $\omega(i+1) \leftarrow \omega$
    If $\delta(\omega(i),\omega) > 0$, $R_k(\omega(i),\omega) \geq \delta(\omega(i),\omega)$, then $\omega(i+1) \leftarrow \omega$
    If $\delta(\omega(i),\omega) > 0$, $R_k(\omega(i),\omega) < \delta(\omega(i),\omega)$, then $\omega(i+1) \leftarrow \omega(i)$
    $n \leftarrow n+1$, $i \leftarrow i+1$
    Until $n = N(k)$
    $n \leftarrow 1$, $k \leftarrow k+1$
  Until $k = K$

All GHC algorithms are formulated using two components, a set of *hill climbing random variables*, $\{R_k\}$, and a neighborhood function, η. This structure permits exploration into the behavior of families of GHC algorithms. GHC algorithms have two iteration counters, an outer loop counter, k, and an inner loop counter, n. The upper bounds, K and N(k), define the algorithm's stopping criteria. The number of iterations for each inner loop is N(k), where K is the number of outer loops. When the stopping criterion of an inner loop is met (i.e., n=N(k)), all inner loop parameters can change (i.e., R_k and N(k)). When the stopping criteria of an outer loop is met (i.e., k=K) the algorithm terminates.

2.4.1 Local Search Algorithms Modeled by Generalized Hill Climbing Algorithms

Numerous local search algorithms can be formulated as particular GHC algorithms by defining different hill climbing random variables. This section defines how to formulate several commonly used local search algorithms for addressing intractable discrete optimization problems as a GHC algorithm.

Several well-known local search algorithms can be defined using the GHC algorithm framework. Monte Carlo search (Johnson and Jacobson 2000b) accepts every neighbor with probability one. Monte Carlo search can be formulated as a GHC algorithm by setting $R_k(\omega(i), \omega)=+\infty$, for all $\omega(i)\in\Omega$, $\omega\in\eta(\omega(i))=\Omega$, and k=1, 2, …, K. Pure local search accepts only neighbors of improving (lower) objective function value. Pure local search can be formulated as a GHC algorithm by setting $R_k(\omega(i), \omega)=0$, for all $\omega(i)\in\Omega$, $\omega\in\eta(\omega(i))$, and k=1, 2, …, K.

Threshold accepting (Dueck and Scheuer 1990) accepts neighbors with higher costs according to a sequence of constants thresholds, Q_k, k=1, 2, …, K. Threshold accepting can be formulated as a GHC algorithm by setting $R_k(\omega(i), \omega)=Q_k$ for all $\omega(i)\in\Omega$, $\omega\in\eta(\omega(i))$, and k=1, 2, …, K, where Q_k is a constant. Simulated annealing (Eglese 1990, Fleischer 1995) accepts neighbors of higher costs with a decreasing probability, where $P\{R_k(\omega(i), \omega)\geq\delta(\omega(i), \omega)\}=e^{\left(-\delta(\omega(i),\omega)/t_k\right)}$ for all $\omega(i)\in\Omega$, $\omega\in\eta(\omega(i))$, and k=1, 2, …, K, for some simulated annealing temperature parameter $t_k>0$, where $\lim_{k\to\infty} t_k=0$. Simulated annealing can be formulated as a GHC algorithm by defining a simulated annealing temperature parameter, t_k, such that $t_k>0$ and $\lim_{k\to\infty} t_k=0$, and setting $R_k(\omega(i), \omega)=-t_k\ln(U)$, for all $\omega(i)\in\Omega$, $\omega\in\eta(\omega(i))$, and k=1, 2, …, K, where U =U(0,1).

Several new local search algorithms can be defined using the GHC algorithm framework. For example, geometric accepting can be formulated as a GHC algorithm by setting $R_k(\omega(i), \omega)= \ln(1-U)/\ln(1-P_k)$, for all $\omega(i)\in\Omega$, $\omega\in\eta(\omega(i))$, and k=1, 2, …, K, where U=U(0,1) and $0<P_k<1$. Weibull accepting is formulated as a GHC algorithm by setting $R_k(\omega(i), \omega)=t_k(-\ln(U))^{1/\alpha}$ for all $\omega(i)\in\Omega$, $\omega\in\eta(\omega(i))$, and k=1, 2, …, K, U=U(0,1), the shape parameter $\alpha>0$ and the scale parameter $t_k>0$. Erlang accepting can be formulated as a GHC algorithm by setting $R_k(\omega(i), \omega)=-t_k(\ln(U_1)+\ln(U_2))$, for all $\omega(i)\in\Omega$, $\omega\in\eta(\omega(i))$, and k=1, 2, …, K, where U_1 and U_2 are independently and identically distributed U(0, 1), and scale parameter $t_k>0$. Normal accepting can be formulated as a GHC algorithm by setting $R_k(\omega(i), \omega)=e_k+(-$

$2\ln(U_1))^{1/2}\ \sin(2\pi U_2)v_k$, for all $\omega(i)\in\Omega$, $\omega\in\eta(\omega(i))$, and k=1, 2, …, K, where U_1 and U_2 are independently and identically distributed U(0, 1), e_k is the mean, and v_k is the standard deviation.

Chapter 3:

Motivational Example

The Materials Process Design Branch of the Air Force Research Laboratory, Wright Patterson Air Force Base (Dayton, Ohio, USA), is faced with the challenge of identifying optimal manufacturing process designs, so that the finished unit meets certain geometric and micro structural specifications, and is produced at minimum cost. To date, the expensive and time intensive approach of trial and error (on the shop floor) has been used to identify feasible manufacturing process designs. The Materials Process Design Branch of the Air Force Research Laboratory, in conjunction with researchers at Ohio University (Athens, Ohio, USA), has developed computer simulation models of manufacturing processes, such as forging and machining. Each such process affects the geometry and/or microstructure of the manufactured unit. Associated with each process are input (controllable and uncontrollable) and output parameters. An exhaustive search through all possible process sequences and controllable input parameters would take a prohibitive amount of time, hence is infeasible. Therefore, it is necessary to construct efficient and effective *optimization algorithms* to identify optimal/near-optimal designs for manufacturers using manufacturing process design computer simulation models in their manufacturing process design planning operations.

Jacobson et al. (1998) describes how to use GHC algorithms in conjunction with computer simulation models of manufacturing processes, to address discrete manufacturing process design optimization problems. The purpose of this chapter is to illustrate how GHC

algorithms can be adapted to identify both optimal controllable input parameters and the optimal valid manufacturing process design sequence (among several possible feasible design sequences). A new neighborhood function is introduced that allows a GHC algorithm to simultaneously optimize over both the controllable input parameters and the design sequences. Computational results are reported with this new neighborhood function.

To describe the discrete manufacturing process design optimization problem and how GHC algorithms can be used to address the problem, Jacobson et al. (1998) present an extensive problem description and GHC algorithm discussion. For completeness, the problem description is presented here (see Section 2.4 for a complete discussion of GHC). First, several definitions are needed.

Let the manufacturing processes be denoted by P_1, P_2, …, P_n. Associated with each process are (continuous or discrete) *controllable input parameters*, *uncontrollable input parameters*, and *output parameters*. The output parameters for a particular process may serve as the uncontrollable input parameters for a subsequent process. A sequence of processes, together with a particular set of input parameters, constitutes a *manufacturing process design*; label such designs D_1, D_2, …, D_N. Note that if one (or more) controllable input parameter is continuous, then $N=+\infty$. Otherwise, $N<+\infty$. Without loss of generality, assume that all the controllable input parameters are discrete, since any continuous controllable input parameter can be discretized over an arbitrarily fine grid. Under this assumption, the number of manufacturing process designs is finite, though potentially very large.

The solution space can be defined as the *design space*, Ω, the set of all manufacturing process designs (i.e., $\Omega=\{D_1, D_2, \ldots, D_N\}$), where a subset of the designs in Ω are feasible. Infeasible designs violate pre-specified constraints on the manufacturing processes and the unit being manufactured, including geometric and micro structural properties, and constraint violations on the output parameters (e.g., the required forging press pressure may not exceed its upper bound limitations). The objective function can now be defined by a *cost function* $f: \Omega \rightarrow [0, +\infty)$ that assigns a non-negative value to each element of the design space, where cost includes monetary costs and costs associated with how well the finished unit meets pre-

specified geometric and micro structural properties. Penalties for constraint violations on the output parameters and measures that ensure a robust manufacturing design (i.e., the manufacturing process design is stable) are included in the cost function. Define a *neighborhood function* $\eta: \Omega \rightarrow 2^{\Omega}$, where $\eta(D) \subset \Omega$ for all $D \in \Omega$. The neighborhood function establishes connections between the designs in the design space (either through the controllable input parameters or through the design sequences), hence allowing the design space to be traversed or searched by moving between designs. For all solutions in the solution space, an individual neighbor can be generated using generation probabilities (i.e., a probability mass function) among all possible neighbors, as defined by the neighborhood function η (Johnson and Jacobson 2000). The goal is to identify the globally optimal manufacturing process design D* (i.e., $f(D^*) \leq f(D)$ for all $D \in \Omega$), or more realistically, near-optimal designs.

3.1 Manufacturing Process Design Application

A manufacturing process design is needed to transform a billet into an integrated blade rotor (IBR) geometric shape (Gunasekera et al. 1996). The possible manufacturing process design sequences are depicted in Figure 3.1.

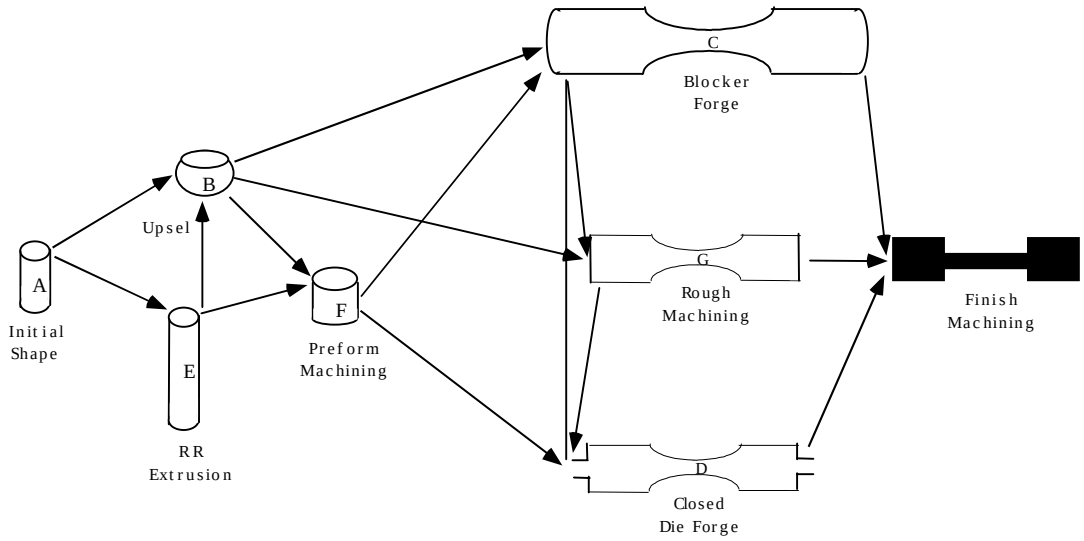

Figure 3.1: Manufacturing Process Design Sequences

Five manufacturing process design sequences have been identified that can achieve this transformation. Define the following notation for the seven processes that make up these five designs:

P_0 is the cast ingot process
P_1 is the extrusion process
P_2 is the upset process
P_3 is the machine preform process
P_4 is the blocker forge process
P_5 is the rough machining process
P_6 is the finished shape process

The five possible manufacturing process design sequences, provided by researchers at the Materials Process Design Branch of the Air Force Research Laboratory and Ohio University, are

$P_0P_2P_4P_6$ $P_0P_2P_5P_6$ $P_0P_2P_3P_4P_6$ $P_0P_1P_2P_4P_6$ $P_0P_1P_3P_4P_6$

Associated with each of the seven processes are uncontrollable and controllable input parameters, and output parameters. For example, for process P_0, there are two controllable input parameters (the radius of the billet and the height of the billet), zero uncontrollable input parameters, and two output parameters, which are just the controllable input parameter values. Fischer et al. (1997) and Gunasekera et al. (1996) present complete details on all seven processes and their parameters.

Associated with each controllable input parameter is a discrete (naturally, or discretized from a continuous domain) set of feasible values. The cost function quantifies the cost associated with not meeting certain geometric and micro structural properties of the finished product, the monetary cost in producing the finished product, and cost penalties for constraint violations (Jacobson et al. 1998). The goal is to identify a valid manufacturing process design sequence, together with values for all the controllable input parameters for the

processes, that result in a feasible manufacturing process design that produces the IBR unit at total minimum cost.

Computer simulation models of the manufacturing processes described above have been developed (Fischer et al. 1997, Gunasekera et al. 1996). This moves the search for an optimal manufacturing process design from the shop floor (where trial and error has typically been applied, using actual materials) to a computer platform. However, even using high speed computing resources, the search for an optimal manufacturing process design may take a prohibitive amount of time. To circumvent this problem, GHC algorithms have been introduced as a tool to be used with the computer simulation models to identify optimal/near-optimal manufacturing process designs.

Jacobson et al. (1998) use three different neighborhood functions for GHC algorithms to identify optimal controllable input parameter values for a set of fixed valid manufacturing process design sequences. By considering each design sequence individually, Jacobson et al. (1998) solves for the controllable input parameter values that minimize the cost. Jacobson et al. (1998) then chooses the overall minimum cost design sequence (and associate optimal controllable input parameter values) as the optimal design. For the five design sequence problem described previously, this is a reasonable approach. However, for extremely complex parts, where the number of design sequences may be very large, performing such optimization (one design sequence at a time) does not lend itself to efficient, automated optimization procedures. Moreover, such an approach does not exploit any common subsequence components within two or more design sequences, which can lead to added efficiencies in identifying optimal controllable input parameter values.

To address these problems, applying GHC algorithms to optimize *between* (or across) design sequences requires a new neighborhood function that captures the cost differences among the different process design sequences, but also allows for transitions (using a well-defined neighborhood function) between such sequences. To define such a neighborhood function requires several new definitions and notations.

Define S to be the set of possible manufacturing process design sequences and W to be the set of possible values for the controllable input parameters for the seven processes. Therefore, each design D can be represented as a two-tuple (**w**, **s**) where $\mathbf{w} \in W$ and $\mathbf{s} \in S$. The neighborhood function is denoted by $\eta(D) = \eta(\mathbf{w}, \mathbf{s}) = (\eta_1(\mathbf{w}), \eta_2(\mathbf{s}))$, where $\eta_1: W \rightarrow 2^W$ and $\eta_2: S \rightarrow 2^S$. The neighborhood function η allows the GHC algorithm to simultaneously change both the input parameters and the manufacturing process design sequence.

There are numerous ways to define the neighborhood functions η_1 and η_2. For the experiments reported, for each $\mathbf{w} \in W$

$$\eta_1(\mathbf{w}) = \{\mathbf{w}' \in W \mid \mathbf{w}' \text{ and } \mathbf{w} \text{ have } \textit{at most} \text{ one controllable input parameter different for each process}\}.$$

To define the neighborhood function η_2 on the set S, the process design sequences can be represented using *binary activity* vectors. Given a process design sequence $\mathbf{s} \in S$, let

$$V_i = \begin{cases} 1, \text{if } P_i \text{ is contained in s} \\ 0, \text{otherwise} \end{cases}.$$

The process design sequence s can be represented by the binary activity vector $\mathbf{s} \in R^5$ where

$$\mathbf{s} = (V_1, V_2, V_3, V_4, V_5).$$

Note that every manufacturing process design sequence begins with P_0 (cast ingot) and ends with P_6 (finish machining), hence a binary activity vector of length five can represent each such sequence. To define such vectors, a strict precedence relation must be imposed on the order in which the processes can occur. For the seven processes, the precedence relation is given as $P_0 <. P_1 <. P_2 <. P_3 <. P_4 <. P_5 <. P_6$, where P' <. P" means process P' must occur before process P" when both P' and P" occur in the same manufacturing process design sequence. Therefore, the set of *possible manufacturing process* design sequences can be formally defined using the binary activity vectors,

$$S = \{\mathbf{s} \in R^5 \mid \mathbf{s} \text{ is a binary activity vector of length five}\}.$$

For the computational results section, S contains thirty-two possible manufacturing process design sequences. Of these, only the five manufacturing process design sequences depicted in the backgrounds section are considered valid. These five manufacturing process design sequences, denoted as $S_v \subseteq S$, are referred to as *valid manufacturing process design sequences*. Table 3-1 depicts these five valid manufacturing process design sequences and their corresponding binary activity vectors.

Table 3-1: Binary Vectors

Binary Activity Vectors for Valid Manufacturing Process Design Sequences	
Process Sequence	Binary Activity Vector
$P_0P_2P_4P_6$	(0,1,0,1,0)
$P_0P_2P_5P_6$	(0,1,0,0,1)
$P_0P_2P_3P_4P_6$	(0,1,1,1,0)
$P_0P_1P_2P_4P_6$	(1,1,0,1,0)
$P_0P_1P_3P_4P_6$	(1,0,1,1,0)

Using the binary activity vector representation for elements of S, the neighborhood function η_2 can be implemented in a GHC algorithm with a random variable that maps elements of S to elements of S. The neighborhood function η_2 depends on a *probability switch* vector. The components of a probability switch vector $\mathbf{q}=(q_1, q_2, q_3, q_4, q_5)$ are the probabilities that a particular process is switched from active (one) to inactive (zero) or from inactive (zero) to active (one). In order to generate a neighbor from $\eta_2(\mathbf{s})$, $\mathbf{s} \in S_v$, the components of the probability switch vector **q** are permuted to the vector **q'**, with the neighbor generated by switching each component of **s**, namely s_i, i=1, 2, 3, 4, 5, with the probability q'_i. Note that if one or more of the q_i, i=1, 2, 3, 4, 5, are zero, the neighborhood function may not contain the entire set of thirty-two possible design sequences. Therefore, for each $\mathbf{s} \in S_v$, setting one or more of the q_i, i=1, 2, 3, 4, 5, to zero can reduce the number of possible manufacturing process design sequences in $\eta_2(\mathbf{s})$.

To switch a process from active to inactive or vice versa, define a set of random variables $\delta_p:\{0,1\} \rightarrow \{0,1\}$ where $p \in [0,1]$ and

$$\delta_p(x) = \begin{cases} x & \text{with probability } 1 - p \\ 1 - x & \text{with probability } p \end{cases}. \qquad (1)$$

Define the *permutation random variable,* Π_j: $[0,1]^5 \rightarrow [0,1]^5$, which permutes the components of a vector in $[0,1]^5$. The permutation random variable will be used to permute the components of a probability switch vector. The permutation random variable is defined so that every permutation **p'** of **p** = (p_1, p_2, p_3, p_4, p_5) occurs with equal probability. Therefore, at iteration j,

$$\Pr\{\Pi_j(\mathbf{p}) = \mathbf{p'}\} = 1 / 5!$$

for all permutations **p'** of **p**, hence a potential neighboring solution of **s** can be obtained using (1) as

$$\mathbf{s'} = (\delta_{p'_1}(s_1), \delta_{p'_2}(s_2), \delta_{p'_3}(s_3), \delta_{p'_4}(s_4), \delta_{p'_5}(s_5)),$$

where $\Pi_j(\mathbf{q}) = \mathbf{q'} = (q'_1, q'_2, q'_3, q'_4, q'_5)$ and **q** is the probability switch vector. Note that if $\mathbf{s'} \notin S_v$, then the neighboring solution of **s** is generated to be **s**.

Given the probability switch vector **q,** let *h* be the number of non-zero elements of **q**. Then neighborhood function, η_2, is defined as

$$\eta_2(\mathbf{s}) = \{\mathbf{s'} \in S \mid \mathbf{s'} \text{ has at most } h \text{ binary components different from } \mathbf{s}\}.$$

Note that for this neighborhood function, the generation probabilities are typically not uniform, since they depend on the probability switch vector.

The choice of probability switch vector **q** depends on $|S|$ (the cardinality of the possible process design sequences) compared to $|S_v|$ (the cardinality of the valid process design sequences). For the manufacturing problem, the number of possible design sequences is 2^5=32, which is large compared to the number of valid process design sequences. There is a trade-off between quickly moving between different manufacturing process design sequences and allowing the GHC algorithm to spend a large number of iterations exploring the various input parameter values in one particular design sequence. A neighborhood function that generates a limited number of design sequence changes may be too restrictive (myopic),

hence prevent the GHC algorithm execution from visiting the (globally) optimal manufacturing process design sequence. This is the same problem that arises with many local search algorithms, namely how to choose the neighborhood function (i.e., should it be myopic, with small neighborhoods, or aggressive, with large neighborhoods.)

3.2 Movement between Valid Manufacturing Process Design Sequences

This section develops a mathematical structure to compute the probability of moving between the five valid manufacturing process design sequences, using the neighborhood function, η_2, introduced in the previous section. To define this structure, let Ψ: [0, 1] $\times$ S $\rightarrow$ S determine a neighbor of **s** (i.e., **s′**) by first permuting **p** (i.e., by applying Π_j(**p**)) and then generating **s′** = $(\delta_{p'_1}(s_1), \delta_{p'_2}(s_2), \delta_{p'_3}(s_3), \delta_{p'_4}(s_4), \delta_{p'_5}(s_5))$. Note that references to an iteration implies an application of Ψ(**p, s**) = s′. Moreover, recall that the possible design sequences are the $32 = |\{0 , 1\}|^5 = 2^5$ elements in the range of Ψ(**p, s**), and that of these 32 possible design sequences, only 5 represent valid manufacturing process design sequences.

To formally define the distance between manufacturing process design sequences, consider the metric space $<\Sigma^n, \rho>$, with $\Sigma = \{0 , 1\}$, where the metric ρ is defined on $\Sigma^n \times \Sigma^n$ such that the *distance* between two possible manufacturing process design sequences **s** = $(s_1, s_2, \ldots, s_n) \in \Sigma^n$ and **t** = $(t_1, t_2, \ldots, t_n) \in \Sigma^n$ is

$$\rho(\mathbf{s}, \mathbf{t}) = |s_1 - t_1| + \ldots + |s_n - t_n| \qquad (2)$$

(Royden, 1988, p.140). To illustrate this metric, the distance between $P_0P_1P_2P_4P_6$ (with activity vector **s** = (1, 1, 0, 1, 0)) and $P_0P_1P_3P_4P_6$ (with activity vector **t** = (1, 0, 1, 1, 0)) is ρ(**s**, **t**) = 2. Define

$$E = \{\mathbf{s} \in \Sigma^n \mid \rho(\mathbf{s}, 0) = 1\},$$

where $\boldsymbol{\varepsilon}^{\mathbf{h}} \in E$ such that $\boldsymbol{\varepsilon}^{\mathbf{h}}$ = (0, …, 0, 1, 0, …, 0) with the one appearing in the h^{th} position.

Define a *binary switch* on the j^{th} element of an activity vector **s** to be the modulus two addition of $\varepsilon^j \in E$ with **s**. For example, consider the activity vector **s** = (1, 1, 0, 1, 0) corresponding to $P_0P_1P_2P_4P_6$. By performing a binary switch on the second element of **s** (i.e., by adding $\boldsymbol{\varepsilon^2}$ = (0, 1, 0, 0, 0)), the resulting activity vector is $\mathbf{s'} = \mathbf{s} + \boldsymbol{\varepsilon^2}$ = (1, 0, 0, 1, 0), corresponding to $P_0P_1P_4P_6$. By performing a second binary switch (i.e., by adding $\boldsymbol{\varepsilon^3}$ = (0, 0, 1, 0, 0) to **s′**), the resulting activity vector is $\mathbf{t} = \mathbf{s'} + \boldsymbol{\varepsilon^3}$ = (1, 0, 1, 1, 0), corresponding to $P_0P_1P_3P_4P_6$. Therefore, the metric $\rho(\mathbf{s}, \mathbf{t})$ represents the minimum number of binary switches required to move from **s** to **t**. For example, the distance between $P_0P_1P_2P_4P_6$ and $P_0P_1P_3P_4P_6$, represented as activity vectors **s** = (1, 1, 0, 1, 0) and **t** = (1, 0, 1, 1, 0), respectively, is $\rho(\mathbf{s}, \mathbf{t})$ = 2. This distance can also be obtained by observing the minimal number of binary switches required to move from **s** to **t**. The distances between all five valid manufacturing process design sequences are given in Table 3-2.

Table 3-2: Distances Between Valid Manufacturing Process Design Sequences

	$P_0P_2P_4P_6$ (0,1,0,1,0)	$P_0P_2P_5P_6$ (0,1,0,0,1)	$P_0P_2P_3P_4P_6$ (0,1,1,1,0)	$P_0P_1P_2P_4P_6$ (1,1,0,1,0)	$P_0P_1P_3P_4P_6$ (1,0,1,1,0)
$P_0P_2P_4P_6$ (0,1,0,1,0)	0	2	1	1	3
$P_0P_2P_5P_6$ (0,1,0,0,1)	2	0	3	3	5
$P_0P_2P_3P_4P_6$ (0,1,1,1,0)	1	3	0	2	2
$P_0P_1P_2P_4P_6$ (1,1,0,1,0)	1	3	2	0	2
$P_0P_1P_3P_4P_6$ (1,0,1,1,0)	3	5	2	2	0

The domain of the neighborhood function η_2 is $S = \{\mathbf{s} \in \Sigma^n \mid \mathbf{s}$ is an activity vector of a valid manufacturing process design sequence$\}$, where $\eta_2(\mathbf{s'})$ (for a fixed $\mathbf{s'} \in S$) is denoted by $\Sigma_i \subseteq \Sigma^n$. For all $\mathbf{s} \in \Sigma_i$, the probability that $\psi(\mathbf{p}, \mathbf{s'}) = \mathbf{s}$ given that $\rho(\mathbf{s'}, \mathbf{s}) = k$ (see (2)) is

$$\Pr\{\psi(\mathbf{p}, \mathbf{s'}) = \mathbf{s}\} = \Pr\{\rho(\mathbf{s'}, \mathbf{s}) = k\} / \binom{5}{k} \text{ for } \mathbf{s} \text{ such that } \rho(\mathbf{s'}, \mathbf{s}) = k.$$

To determine Pr{$\rho(\mathbf{s'}, \mathbf{s})$=k}, recall that $\psi(\mathbf{p}, \mathbf{s'})$=**s** is a sequence of binary switches on **s′**, where the cardinality of binary switches (the previously defined distance) is dependent on **p** = (p_1, p_2, p_3, p_4, p_5). Table 3-3 provides the probabilities that $\rho(\mathbf{s'}, \mathbf{s})$=k for k=1, 2, 3, 4, 5.

To illustrate the computation of these probabilities, if η_2 is defined with **p**=(.5, 0, 0, 0, 0), then $\psi(\mathbf{p}, \mathbf{s'})$ results in no binary switches with probability .5, and one binary switch with probability .5. If η_2 is defined with **p**=(.5, .5, 0, 0, 0), then $\psi(\mathbf{p}, \mathbf{s'})$ results in no binary switches with probability .25, one binary switch with probability .5, and two binary switches with probability .25. The five valid manufacturing process design sequences can be traversed using **p**=(p_1, p_2, 0, 0, 0). For neighborhood function η_2, with **p**=(p_1, p_2, 0, 0, 0), it is possible to move between valid design sequences (in a single iteration) with activity vectors that are at most two binary switches apart. Figure 3.2 depicts movement between sequences in a single iteration, where the nodes represent the five valid manufacturing process design sequences, and the values on the edges represent the distance between the sequences at the corresponding nodes.

Table 3-3: Distance Probabilities

k	Pr{$\rho(\mathbf{s'}, \mathbf{s})$ = k}
1	$\sum_{j=1}^{5} p_j \prod_{\substack{i=1 \\ i\neq j}}^{5} (1-p_i)$
2	$\sum_{i=1}^{5} p_j \sum_{\substack{j=1 \\ j\neq i}}^{5} p_j \prod_{\substack{k=1 \\ k\neq j \\ k\neq i}}^{5} \frac{(1-p_k)}{2}$
3	$\sum_{i=1}^{5} (1-p_j) \sum_{\substack{j=1 \\ j\neq i}}^{5} (1-p_j) \prod_{\substack{k=1 \\ k\neq j \\ k\neq i}}^{5} \frac{p_k}{2}$
4	$\sum_{j=1}^{5} (1-p_j) \prod_{\substack{i=1 \\ i\neq j}}^{5} p_i$
5	$p_1 p_2 p_3 p_4 p_5$

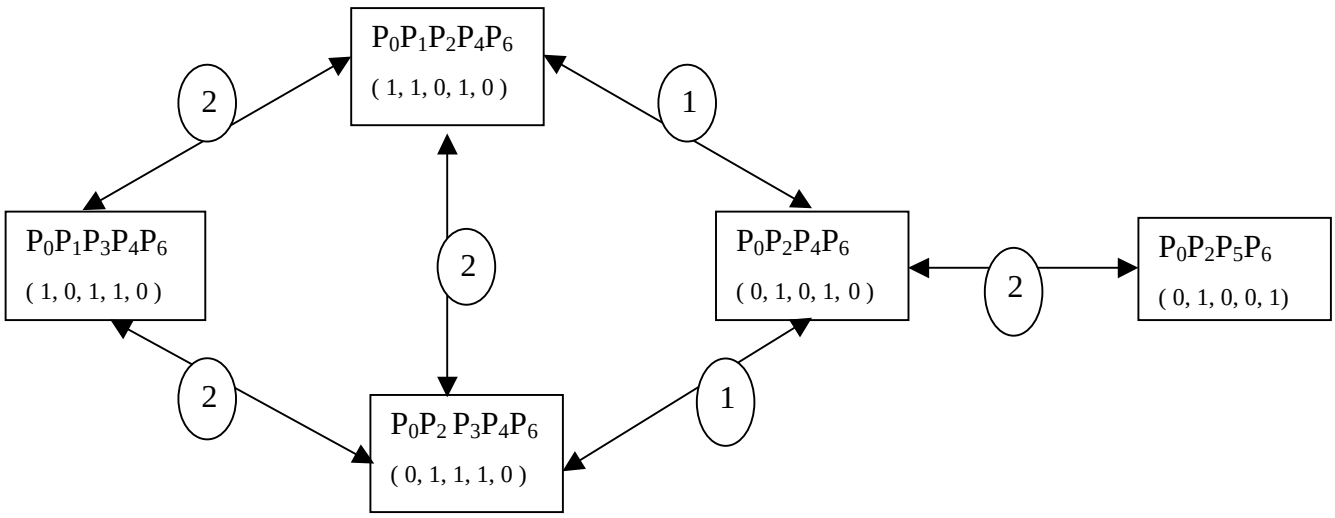

Figure 3.2: The Five Valid Manufacturing Process Design Sequences

For the probability switch vector **p** = (p_1, p_2, 0, 0, 0), where $0 < p_1, p_2 < 1$, the probability of moving between sequences at any given iteration is a function of the number of binary switches required to move between these sequences. In particular, define the two *travel probabilities*

$$\Pr\{\text{1-binary switch}\} = \left(\frac{p_1(1-p_2)}{\binom{5}{1}} + \frac{p_2(1-p_1)}{\binom{5}{1}}\right) = \frac{(p_1(1-p_2)+p_2(1-p_1))}{5}$$

and

$$\Pr\{\text{2-binary switches}\} = \frac{p_1 p_2}{\binom{5}{2}} = \frac{p_1 p_2}{10}.$$

These expressions are validated empirically using Monte Carlo search, by recording the number of times the algorithm iterated from a source design sequence (say A) to a neighboring design sequence (say B) and then dividing by the number of times the algorithm visited the source design sequence (A). Table 3-4 provides the Monte Carlo search computational results for **p**=(p_1, p_2, 0, 0, 0, 0)=(.9, .9, 0, 0, 0), with 100,000 iterations. Note that for Monte Carlo search, all iterations are outer loop iterations with one inner loop iteration per outer loop iteration (i.e., K=100,000 and N(k)=1, k=1, 2, …, 100,000). Ninety

percent confidence intervals (CI) are reported for the travel probabilities, where the confidence intervals marked with an asterisk are those that cover the true travel probability values (of these twelve confidence intervals, nine cover the true travel probability value).

Table 3-4: Traveling Between Sequences

P = (.9, .9, 0, 0, 0)					
Sequence A	Sequence B	Number of Binary Switches	True Travel Probability	Estimated Travel Probability	True Travel Probability CI
$P_0P_1P_3P_4P_6$ (1, 0, 1, 1, 0)	$P_0P_1P_2P_4P_6$ (1, 1, 0, 1, 0)	2	.081	.079	(.0769, .0811)*
$P_0P_1P_3P_4P_6$ (1, 0, 1, 1, 0)	$P_0P_2P_3P_4P_6$ (0, 1, 1, 1, 0)	2	.081	.083	(.0809, .0851)*
$P_0P_1P_2P_4P_6$ (1, 1, 0, 1, 0)	$P_0P_1P_3P_4P_6$ (1, 0, 1, 1, 0)	2	.081	.076	(.0728, .0792)
$P_0P_1P_2P_4P_6$ (1, 1, 0, 1, 0)	$P_0P_2P_3P_4P_6$ (0, 1, 1, 1, 0)	2	.081	.078	(.0748, .0812)*
$P_0P_1P_2P_4P_6$ (1, 1, 0, 1, 0)	$P_0P_2P_4P_6$ (0, 1, 0, 1, 0)	1	.036	.038	(.0358, .0402)*
$P_0P_2P_3P_4P_6$ (0, 1, 1, 1, 0)	$P_0P_1P_3P_4P_6$ (1, 0, 1, 1, 0)	2	.081	.078	(.0748, .0812)*
$P_0P_2P_3P_4P_6$ (0, 1, 1, 1, 0)	$P_0P_1P_2P_4P_6$ (1, 1, 0, 1, 0)	2	.081	.082	(.0788, .0852)*
$P_0P_2P_3P_4P_6$ (0, 1, 1, 1, 0)	$P_0P_2P_4P_6$ (0, 1, 0, 1, 0)	1	.036	.034	(.0318, .0362)*
$P_0P_2P_4P_6$ (0, 1, 0, 1, 0)	$P_0P_1P_2P_4P_6$ (1, 1, 0, 1, 0)	1	.036	.033	(.0309, .0351)
$P_0P_2P_4P_6$ (0, 1, 0, 1, 0)	$P_0P_2P_5P_6$ (0, 1, 0, 0, 1)	2	.081	.082	(.0789, .0851)*
$P_0P_2P_4P_6$ (0, 1, 0, 1, 0)	$P_0P_2P_3P_4P_6$ (0, 1, 1, 1, 0)	1	.036	.034	(.0319, .0361)*
$P_0P_2P_5P_6$ (0, 1, 0, 0, 1)	$P_0P_2P_4P_6$ (0, 1, 0, 1, 0)	2	.081	.089	(.0858, .0922)

3.3 Computational Results

Computational results are reported with the neighborhood function that allows for optimization between the manufacturing process design sequences. The computational results in Jacobson et al. (1998) suggest that the Weibull accepting hill climbing random

variable was the most effective (among five different GHC algorithm formulations tested) for optimizing over the controllable input parameters for a particular design sequence, hence it was used to define η_1. The objective in running these experiments is to assess the performance of the GHC algorithm formulations using the neighborhood functions (η_1, η_2), as well as to identify optimal/near optimal manufacturing process design sequences and input parameters using computer simulation models.

For Weibull accepting, t_k is updated by multiplying the previous temperature parameter by the increment multiplier β_1, where $0 \leq \beta_1 \leq 1$ (i.e., $t_k = \beta_1 t_{k-1}$). The initial temperature parameter is $t_0 = 10{,}000$, with $\beta_1 = .96$ and shape parameter $\alpha = 2.0$. The acceptance probability for the Weibull accepting GHC algorithm is

$$\Pr\{R_k(D, D') \geq \delta\} = e^{(-\delta / t_k)^{\alpha}}, \; D \in \Omega, \; D' \in \eta(D) \text{ for all } k=1, 2, \ldots, K. \qquad (3)$$

Note that if $\alpha=1$, then Weibull accepting reduces to simulated annealing (Jacobson et al. 1998).

The cost function evaluates the total cost associated with the simulated manufacturing process design, in US dollars. The initial cost is the cost of the initial billet, which depends on the dimensions of the billet and the specific metal being processed. The costs for the forging processes include:

i) set up costs,

ii) post-inspection costs,

iii) die wear costs,

iv) press run costs,

v) the cost of possible strain-induced-porosity damage in the work piece.

Penalties are incurred with the forging processes when

i) the press capacity is exceeded,

ii) the aspect ratio of the work piece is too large,

iii) the geometry of the work piece conflicts with the die geometry.

The cost to machine the work piece is the cost of the material removed from the work piece, where a penalty cost is incurred when the geometry of the work piece conflicts with the desired final geometry of the work piece after machining. After the work piece is processed, a mandatory ultrasonic non-destructive evaluation cost and, if necessary, a cost of heat treatment is accrued. In addition, the final microstructure of the work piece is evaluated; if the microstructure violates predetermined specifications, a penalty cost is incurred. All penalties are translated into US dollars in the cost function.

Computational results with the Weibull accepting GHC algorithm incorporating the new neighborhood function (η_2) are reported. The Weibull accepting GHC algorithms were executed with different values of K and N=N(k), k=1, 2, …, K, as well as with varying values for the components of the probability switch vector **p**. Thirty (independently seeded) replications of each GHC formulation were made, each initialized with a different initial manufacturing process design sequence. The same thirty initial design sequences (for the thirty replications) were used across the different neighborhood functions (i.e., the different probability switch vectors). The initial controllable input parameter values for replications two through thirty were obtained by randomly selecting a neighbor (η_1) of the first replication's feasible controllable input parameters values. The mean (μ) and standard deviations (σ), as well as the minimum and maximum cost function values, were computed from the optimal values across these thirty replications. All computational experiments were executed on a SUN ULTRA-1 workstation (128 Mb RAM). Each set of thirty replications took approximately 30 CPU minutes.

In Tables 3-5 through 3-7, the variable γ represents the percentage of the replications that the GHC algorithm finds the optimal valid manufacturing process design sequence. The variable κ represents the average number of times, over the thirty replications, neighborhood function η_2 generates a manufacturing process design sequence that is different from the incumbent design sequence.

Table 3-5: GHC Algorithm Results

K = 50 and N = 200

Probability Switch Vector	γ	μ	σ	Minimum	Maximum	κ
(0.1, 0.1, 0, 0, 0)	9/30	2177.57	199.00	1935.71	2928.17	357.7
(0.2, 0.2, 0, 0, 0)	22/30	2047.62	209.89	1927.01	2928.72	538.6
(0.3, 0.3, 0, 0, 0)	20/30	2041.25	145.93	1919.28	2250.15	793.1
(0.4, 0.4, 0, 0, 0)	27/30	1973.88	92.00	1930.43	2245.28	1001.5
(0.5, 0.5, 0, 0, 0)	21/30	2033.21	140.17	1919.28	2248.08	1110.7
(0.6, 0.6, 0, 0, 0)	21/30	2033.88	142.08	1919.28	2254.63	1268.8
(0.7, 0.7, 0, 0, 0)	24/30	2009.45	118.78	1921.21	2250.15	1418.1
(0.75, 0.75,0, 0, 0)	25/30	1998.00	111.67	1927.01	2250.15	1408.2
(0.8, 0.8, 0, 0, 0)	25/30	1993.57	113.97	1921.21	2245.28	1438.4
(0.833, 0.833, 0, 0, 0)	27/30	1976.08	92.32	1927.01	2245.28	1402.4
(0.9, 0.9, 0, 0, 0)	27/30	1977.57	91.74	1921.21	2250.15	1443.1

Table 3-6: GHC Algorithm Results

K = 100 and N = 100

Probability Switch Vector	γ	μ	σ	Minimum	Maximum	κ
(0.1, 0.1, 0, 0, 0)	15/30	2089.41	156.13	1921.21	2245.28	389.3
(0.2, 0.2, 0, 0, 0)	15/30	2115.83	213.35	1921.21	2898.11	653.9
(0.3, 0.3, 0, 0, 0)	24/30	2000.99	125.00	1921.21	2248.08	800.9
(0.4, 0.4, 0, 0, 0)	25/30	1996.43	112.76	1919.28	2249.97	1133.6
(0.5, 0.5, 0, 0, 0)	25/30	1993.50	113.54	1921.21	2245.28	1285.8
(0.6, 0.6, 0, 0, 0)	22/30	2025.29	132.74	1923.70	2245.28	1389.0
(0.7, 0.7, 0, 0, 0)	24/30	2005.69	120.13	1932.86	2248.08	1498.0
(0.75, 0.75,0, 0, 0)	22/30	2024.25	135.71	1921.21	2250.15	1419.7
(0.8, 0.8, 0, 0, 0)	24/30	2004.95	121.77	1919.28	2244.25	1425.1
(0.833, 0.833, 0, 0, 0)	24/30	2008.17	120.15	1921.21	2245.28	1451.0
(0.9, 0.9, 0, 0, 0)*	27/30	1974.30	93.37	1921.21	2248.08	1397.9

Table 3-7: GHC Algorithm Results

K = 200 and N = 50

Probability Switch Vector	γ	μ	σ	Minimum	Maximum	κ
(0.1, 0.1, 0, 0, 0)	9/30	2196.01	249.69	1919.28	2959.29	344.7
(0.2, 0.2, 0, 0, 0)	11/30	2131.19	154.88	1923.15	2293.87	714.1
(0.3, 0.3, 0, 0, 0)	20/30	2037.88	149.32	1919.28	2257.53	799.7
(0.4, 0.4, 0, 0, 0)	22/30	2012.43	141.59	1919.28	2257.53	934.5
(0.5, 0.5, 0, 0, 0)	22/30	2016.57	138.44	1921.21	2245.28	1294.2
(0.6, 0.6, 0, 0, 0)	21/30	2027.38	143.96	1919.28	2248.08	1368.6
(0.7, 0.7, 0, 0, 0)	23/30	2009.21	132.73	1919.28	2248.08	1531.3
(0.75, 0.75, 0, 0, 0)	21/30	2025.96	143.54	1919.28	2245.28	1439.1
(0.8, 0.8, 0, 0, 0)	20/30	2038.75	147.56	1919.28	2248.08	1424.8
(0.833, 0.833, 0, 0, 0)	17/30	2067.71	154.65	1919.28	2248.08	1435.3
(0.9, 0.9, 0, 0, 0)	19/30	2049.26	152.23	1919.28	2260.63	1403.8

The results in Tables 3-5 through 3-7 illustrate the performance of the Weibull accepting GHC algorithm with the new neighborhood function. In particular, the results demonstrate differences when the values of K, N, and components of the probability switch vector are changed. The Weibull accepting GHC algorithm using the probability switch vectors (.1, .1, 0, 0, 0), (.2, .2, 0, 0, 0), (.3, .3, 0, 0, 0) with $\eta_2(\mathbf{s})$ resulted in a low probability of generating a manufacturing process design sequence other than **s** (thereby providing a myopic neighborhood structures), hence yielded inferior results than those obtained with the probability switch vectors containing higher probability components. This observation leads to the initial conclusion that as the number of manufacturing process design sequence changes increased, the mean value of the optimal solution found by the Weibull accepting algorithm improved.

The results in Tables 3-5 through 3-7 also suggest that choosing probability switch vectors that maximize process design sequence changes can also yield poor solutions, as measured by μ. This results from the algorithm moving too quickly out of optimal manufacturing process design sequences to non-optimal manufacturing process design sequences. Furthermore, the performance of a GHC algorithm depends on the values of K and N. Comparing the values for γ, μ, and σ in Tables 3-5 and 3-6, across the same probability switch vectors values in Tables 3-7, the Weibull accepting GHC algorithm with K=50, N=200 or K=100, N=100 yielded results that are significantly better than for the case with

K=200, N=50 in Table 3-7. Moreover, the Weibull accepting GHC algorithm with K = 50, N=200 or K=100, N=100 also increased the number of hill climbing solutions accepted during the execution of the algorithm. Examining equation (3), at each iteration of the Weibull accepting GHC algorithm, the probability that a hill climbing solution is accepted is large when the value of t_k is large. Therefore, using K=50, N=200, the temperature parameter, t_k, for the Weibull accepting GHC algorithm converges to zero at a sufficiently slow rate such that the probability of accepting a hill climbing solution also decreases very slowly at each outer loop iteration. This results in an increased frequency that the algorithm execution visits the optimal sequence at the beginning of the execution. Visiting the optimal sequence early in the execution of the algorithm also decreases the probability of becoming locked in a non-optimal process design sequence at the end of the algorithm execution.

Note that as a base case, to compare the effectiveness and value of the Weibull accepting algorithm with the probability switch vector neighborhood function, experiments were run with a Weibull accepting algorithm using a neighborhood function (termed the base case neighborhood function) that assigns an integer value (i.e., 1, 2, 3, 4, 5) to each of the five possible manufacturing process design sequences, where the probability of moving from a design sequence to any other design sequence was .25. This algorithm was also executed using the same three combinations of values for K and N=N(k), k=1, 2, …, K. Thirty (independently seeded) replications of each GHC formulation were made. The mean (μ) and standard deviations (σ), as well as the minimum and maximum cost function values, were computed from the optimal values across these thirty replications. These results are given in Table 3-8.

Table 3-8: GHC Algorithm Results

GHC Algorithm Results, Base Case

Inner and Outer Loop Bounds	γ	μ	σ	Minimum	Maximum
K=50, N=200	24/30	2048.95	193.01	1919.28	2723.33
K=100, N=100	25/30	2147.12	623.76	1927.01	5200.51
K=200, N=50	19/30	2201.76	599.32	1924.94	5095.13

The results in Table 3-8, when compared to the results in Tables 3-5 through 3-7, suggest that using the Weibull accepting algorithm with probability switch vector neighborhood function can be more effective in identifying both the optimal design sequence and the optimal controllable input parameter values compared to the Weibull accepting algorithm using base case neighborhood function. Moreover, the variance of the optimal design sequences obtained using the probability switch vector neighborhood function is significantly lower than for the base case neighborhood function. These results can be explained by noting that the base case neighborhood function selects neighboring design sequences uniformly, while the probability switch vector neighborhood function weights this selection based on common manufacturing processes between the design sequences. This overlap provides a more effective strategy in moving between different design sequences, in search of the optimal design sequence.

Overall, the computational results are consistent with what would have been obtained using trial and error on the shop floor. In particular, the optimal design ($P_0P_2P_5P_6$) required smaller initial billets (hence there was less material wasted) and used machining processes (rather than forging and extrusion processes) to achieve the desired shape and size. These results are also consistent with those reported in Jacobson et al. (1998), in terms of the minimum cost solutions obtained. The advantage of using GHC algorithms and computer simulation manufacturing processes is the speed and efficiency at which these results can be obtained, at a fraction of the cost that would be spent if trial and error on the shop floor would be required. Moreover, allowing a GHC algorithm to optimize over both the design sequences and the controllable input parameters provides an important first step to developing automated procedures for such optimization problems.

Chapter 4:

Simultaneous Generalized Hill Climbing Algorithms

Chapter 3 introduced a new neighborhood function for simultaneously addressing a set of related manufacturing process design optimization problems simultaneously using GHC algorithms. This neighborhood function allowed for simultaneous optimization across the design sequences and the controllable input parameters. The application of such optimization algorithms (that simultaneously explore multiple manufacturing process designs) using computer simulation is a new advance in how optimal manufacturing process designs can be efficiently identified (Vaughan et al. 2000).

It is common to encounter several discrete optimization problems where a relationship between the solution spaces of the individual problems exists. This chapter relaxes the methodology used to address the integrated blade rotor discrete manufacturing process design problem to develop a general mathematical framework for simultaneously approaching a set of related discrete optimization problems. The resulting framework is termed *simultaneous generalized hill climbing* (SGHC) algorithms.

4.1 Characterizing Sets of Discrete Optimization Problems

This section formally defines a class of sets of discrete optimization problems where a relationship exists that is similar to the one described for the manufacturing problem described in Chapter 3. A set of discrete optimization problems that is contained in this class is referred to as a set of *fundamentally related* discrete optimization problems. Additionally, this section develops a metric (termed the *detachment metric*) between elements in a set of fundamentally related discrete optimization problems (Vaughan et al. 2000). The detachment metric is a tool for determining if it is advantageous to address a particular set of discrete optimization problems with a SGHC algorithm.

To discuss the class of sets of discrete optimization problems for which SGHC algorithms are applicable, the following definitions are needed. Consider a *set of discrete optimization problems* $S=\{D_1, D_2, \ldots, D_m\}$, where each discrete optimization problem $D_y=(\Omega_y, f_y)$ is fully defined by a finite set of solutions Ω_y and a real-valued objective function $f_y: \Omega_y \rightarrow R$. A set of discrete optimization problems S is *fundamentally related* by a set $Ob=\{c_1, c_2, \ldots, c_n\}$ of objects if the solution space Ω_y of each discrete optimization problem $D_y=(\Omega_y, f_y) \in S$ can be fully defined by exactly one subset of *Ob*. Then for every discrete optimization problem $D_y=(\Omega_y, f_y) \in S$, there is exactly one set $C_y \subseteq Ob$ such that C_y completely defines Ω_y. The set C_y is defined to be the *fundamental relation set* of D_y.

Let S be a set of fundamentally related discrete optimization problems related by *Ob*. Consider $D_y \in S$ where $C_y \subseteq Ob$ is the fundamental relation set of D_y. Then C_y can be represented by the *binary activity vector* $\mathbf{c}^y \in \{0, 1\}^n$, $\mathbf{c}^y=(B_1, B_2, \ldots, B_n)$, where

$$B_i = \begin{cases} 1, & \text{if } c_i \text{ is contained in } C_y \\ 0, & \text{otherwise} \end{cases}.$$

SGHC algorithms are developed for sets of fundamentally related discrete optimization problems. When two discrete optimization problems, D_y and D_q, are contained in a set of fundamentally related discrete optimization problems with respective fundamental relation

sets, C_y and C_q, where $|C_y \cap C_q|/|Ob|$ is close to one, it is reasonable to conjecture that the optimal/near optimal solutions of D_y and D_q are similar. The following detachment metric is designed to determine if two discrete optimization problems, in a set of fundamentally related discrete optimization, are close together, hence have similar solution spaces.

Let S be a set of fundamentally related discrete optimization problems related by *Ob*. To formally define the detachment metric ρ between discrete optimization problems D_y, $D_q \in S$, consider the metric space $<\Sigma^n, \rho>$, with $\Sigma = \{0, 1\}$, where the detachment metric ρ is defined on $\Sigma^n \times \Sigma^n$ such that the *distance* between two discrete optimization problems can be determined by considering their binary activity vectors $\mathbf{c}^{\mathbf{y}}=(c_1^y, c_2^y, ..., c_n^y) \in \Sigma^n$ and $\mathbf{c}^{\mathbf{j}}=(c_1^q, c_2^q, ..., c_n^q) \in \Sigma^n$. Define the *detachment metric* as

$$\rho(D_y, D_q) = \left|c_1^y - c_1^q\right| + \left|c_2^y - c_2^q\right| + ... + \left|c_n^y - c_n^q\right|$$

(Royden, 1988, p.140). This metric is illustrated in Section 3.2. The detachment metric provides a way to measure the overlap (or lack of overlap) between the members in a set of fundamentally related discrete optimization problems.

4.2 Neighborhood Function

This section develops the neighborhood function with an associated problem generation probability function for moving between discrete optimization problems during an execution of a SGHC algorithm. The neighborhood function is defined such that each discrete optimization problem has the entire set of discrete optimization problems as neighbors. Therefore, whenever this neighborhood function is applied, every discrete optimization problem is a *candidate problem* (i.e., has a positive probability of being selected). The *problem generation probability function* determines the probability of selecting a candidate problem.

More formally, define the neighborhood function, $\eta_{set}:S\rightarrow 2^S$, such that $\eta_{set}(D_y)=S$, for all $D_y\in S$. Define the *problem generation probability function* $h_{D_yD_q}(k, \rho(D_y, D_q))$, such that

$$0 < h_{D_yD_q}(k, \rho(D_y, D_q)) < 1, \text{ for every } D_y\in S,\ D_q\in \eta_{set}(D_y),$$

where,

$$\sum_{D_q\in\eta_{set}(D_y)} h_{D_yD_q}(k, \rho(D_y, D_q)) = 1, \text{ for every } D_y\in S,\ D_q\in \eta_{set}(D_y),$$

for every k=1, 2, …, K.

Note that the problem generation probability function (the probability of selecting a candidate problem, $D_q\in \eta_{set}(D_y)$, $D_y\in S$) can be a function of both the outer loop iteration k=1, 2, …, K and the detachment metric $\rho(D_y, D_q)$.

4.3 The Simultaneous Generalized Hill Climbing Algorithm Pseudo-Code

SGHC algorithms provide a mathematical framework for addressing several fundamentally related discrete optimization problems simultaneously using GHC algorithms. SGHC algorithms seek to find optimal solutions for sets of fundamentally related discrete optimization problems by allowing the algorithm to probabilistically move between discrete optimization problems. When a new discrete optimization problem is generated, an initial solution for this new problem is then generated using information from the previous discrete optimization problem's final solution. The inner and outer loop structure defined for GHC algorithms can be used in SGHC algorithms, where SGHC algorithms restrict possible movement between discrete optimization problems to the first iteration of the outer loop iterations. This constraint ensures that a GHC algorithm is applied to each discrete optimization problem at least N(k) iterations each time it is generated (i.e., initially visited). Note that this was not the case for the manufacturing problem presented in Chapter 3, where movement between discrete optimization problems was possible during all inner loop iterations.

The SGHC algorithm pseudo-code is now presented.

Figure 4.1: SGHC Algorithm Pseudo-Code

Set the outer loop counter bound K and the inner loop counter bounds N(k), k=0,1,2,…,K
Define a set of hill climbing (random) variables R_k: $\Omega \times \Omega \rightarrow \Re \cup \{-\infty,+\infty\}$, k=1,2,…,K
Set the iteration indices N(0)=i=0, k=n=1
Select an initial discrete optimization problem $D(0) \in S$
Select an initial solution $\omega(0,0) \in \Omega(0)$
Repeat while $k \leq K$
 Generate a discrete optimization problem $D(k) \in \eta_{set}(D(k-1))$
  If $D(k) \neq D(k-1)$,
   Generate a solution $\omega \in \Omega(k)$ and $\omega(k, 1) \leftarrow \omega$ (new discrete optimization problem)
  else $\omega(k, 1) \leftarrow \omega(k-1, N(k-1))$ (same discrete optimization problem)
  Repeat while $n \leq N(k)$
   Generate a solution $\omega \in \eta(\omega(k, i))$
   Calculate $\delta(\omega(k, i),\omega) = f(\omega)-f(\omega(k, i))$
   If $\delta(\omega(k, i),\omega) \leq 0$, then $\omega(k, i+1) \leftarrow \omega$
   If $\delta(\omega(k, i),\omega) > 0$ and $R_k(\omega(k, i),\omega) \geq \delta(\omega(k, i),\omega)$, then $\omega(k, i+1) \leftarrow \omega$
   If $\delta(\omega(k, i),\omega) > 0$ and $R_k(\omega(k, i),\omega) < \delta(\omega(k, i),\omega)$, then $\omega(k, i+1) \leftarrow \omega(k, i)$
   $n \leftarrow n+1$, $i \leftarrow i+1$
  Until $n = N(k)$
 $n \leftarrow 1$, $k \leftarrow k+1$
Until $k = K$

All SGHC algorithms are formulated using three components, a set of hill climbing random variables $\{R_k\}$, a neighborhood function η between solutions and a neighborhood function η_{set} between discrete optimization problems. The two-tuple (k, i) represents the inner loop iteration i=1, 2, …, N(k), during outer loop iteration k=1, 2, …, K. D(k) is the discrete optimization problem the algorithm is executing over during the k^{th} outer loop iteration, k=1, 2, …, K, where the solution space of D(k) is depicted by $\Omega(k)$.

Chapter 5:

Simultaneous Generalized Hill Climbing Markov Chain Theory

Markov chain theory is an effective tool for studying the performance of local search algorithms. This chapter shows that an application of the SGHC algorithm can be modeled using Markov chains. In particular, Section 5.1 demonstrates that an application of the GHC algorithm can be modeled with a Markov chain. Section 5.2 shows that an application of the SGHC algorithm can be modeled by a set of Markov Chains.

5.1 Generalized Hill Climbing Markov Chain Theory

To show that an application of the GHC algorithm can be modeled with a Markov chain, the following definitions are needed. A *stochastic process* is a family of random variables defined on some state space. If there are countable many members of the family, the process (termed a *discrete-time process*) is denoted by Q_1, Q_2, … where the set of distinct values assumed by a stochastic process is the *state space*. If the state space is countable or finite, the process is a *chain*. A stochastic process $\{Q_k\}$, k=1, 2, … with state space $\Omega=\{\omega_1, \omega_2, \ldots\}$ satisfies the *Markov property* if for every n and for all states $\omega_1, \omega_2, \ldots, \omega_n$

$$\Pr\{Q_n=\omega_n \mid Q_{n-1}=\omega_{n-1}, Q_{n-2}=\omega_{n-2}, \ldots, Q_1=\omega_1\} = \Pr\{Q_n=\omega_n \mid Q_{n-1}=\omega_{n-1}\} = P_{n(n-1)}.$$

A discrete-time stochastic process that satisfies the Markov property is a *Markov chain*.

Let $\{Q_k\}$ denote a discrete-time Markov chain with finite solution space $\Omega=\{\omega_1, \omega_2, \ldots, \omega_{|\Omega|}\}$. For this chain there are $|\Omega|^2$ transition probabilities, $\{P_{ij}\}$, i,j=1, 2, …, $|\Omega|$. The *transition matrix* associated with the Markov chain $\{Q_k\}$ is P, where P_{ij} is the probability of moving from state ω_i to state ω_j.

An application of a GHC algorithm can be modeled by a stochastic process $\{Q_n^k\}$, k=1, 2, …, K, n=1, 2, …, N(k), $Q_n^k \in \Omega$ with solution space $\Omega=\{\omega_1, \omega_2, \ldots, \omega_{|\Omega|}\}$ that satisfies the Markov property for every n and all states $\omega_1, \omega_2, \ldots, \omega_n$ (i.e., $\{Q_n^k\}$ is a Markov chain). To see this, consider an application of a GHC algorithm to a discrete optimization problem with solution space $\Omega=\{\omega_1, \omega_2, \ldots, \omega_{|\Omega|}\}$. Define $g_{ij}(k)$ to be the *generation probability function* for the neighborhood function η, where the probability that $\omega_j \in \eta(\omega_i)$ is generated during outer loop iteration k, is $g_{ij}(k)$. Consider the inner loop iterations for fixed outer loop iteration k=1, 2, …, K. Let $\{Q_n^k\}$, k=1, 2, …, K, n=1, 2, …, N(k), $Q_n^k \in \Omega$ be the stochastic process where if $Q_n^k=\omega_i$, then the GHC algorithm is at solution ω_i during inner loop iteration n and outer loop iteration k (Johnson and Jacobson 2000b). If the GHC algorithm is at solution ω_i at inner loop iteration n-1, the probability that the algorithm is at solution ω_j at inner loop iteration n is

$$P_{ij}(k) = \begin{cases} g_{ij}(k)\Pr(R_k(\omega_i,\omega_j)) \geq \delta_{ij} & \text{for all } \omega_i \in \Omega, \omega_j \in \eta(\omega_i),\ j \neq i \\ 1 - \sum_{\substack{z \in \eta(\omega_i) \\ z \neq i}} P_{iz}(k) & j = i \\ 0 & \text{otherwise} \end{cases},$$

independent of the solutions the algorithm visited at inner loop iterations 1, 2, …, n-2. Therefore, the Markov property holds,

$$\Pr\{Q_n^k=\omega_j \mid Q_{n-1}^k=\omega_i,\ Q_{n-2}^k=\omega_{i\,(n-2)}, \ldots,\ Q_1^k=\omega_{i\,(1)}\} = \Pr\{Q_n^k=\omega_j \mid Q_{n-1}^k=\omega_i\} = P_{ij}(k).$$

Moreover, for every outer loop iteration k, the Markov chain $\{Q_n^k\}$ has a transition matrix P(k), where $P_{ij}(k)$ is defined as above .

5.2 Simultaneous Generalized Hill Climbing Markov Chain Theory

Recall, that a SGHC algorithm is applied to a set of fundamentally related discrete optimization problems. Movement between discrete optimization problems is only possible at outer loop iterations k=1, 2, …, K. During the inner loop iterations, the SGHC algorithm is executing over the solution space of the current discrete optimization problem using a GHC algorithm. Section 5.1 shows that an application of a GHC algorithm can be modeled by a Markov chain.

This section shows that for fixed outer loop iteration k=1, 2, …, K, the stochastic process corresponding to the solution that the SGHC algorithm is at during inner loop iterations n=1, 2, …, N(k) can be modeled by a Markov chain that corresponds to an application of a GHC algorithm. Moreover, it is shown that for outer loop iterations k=1, 2, …, K, the possible movement between discrete optimization problems is a stochastic process that satisfies the Markov property.

Consider an application of a SGHC algorithm to a set of fundamentally related discrete optimization problems $S=\{D_1, D_2, \ldots, D_m\}$, where each discrete optimization problem D_y, y=1, 2, …, m is fully defined by a solution space Ω_y and an objective function f_y (i.e., $D_y=(\Omega_y, f_y)$). Consider the inner loop iterations n=1, 2, …, N(k), for fixed outer loop iteration k, k=1, 2, …, K. Let $\{Q_n^k(D_y)\}$, k=1, 2, …, K, n=1, 2, …, N(k) be the stochastic process where if $Q_n^k(D_y)=\omega_i$, then the SGHC algorithm is at solution $\omega_i \in \Omega_y$ at inner loop iteration n of outer loop iteration k.

Note that, for all inner loop iterations n=1, 2, …, N(k) of an outer loop iteration k=1, 2, …, K the SGHC algorithm is executing over a particular discrete optimization problem from the set

of fundamentally related discrete optimization problems $S=\{D_1, D_2, ..., D_m\}$ using a GHC algorithm. Section 5.1 showed that any application of a GHC algorithm to a discrete optimization problem can be modeled as a Markov chain. Therefore, for fixed outer loop iteration k, the stochastic processes $\{Q_n^k(D_y)\}$, y=1, 2, …, m, with transition matrices P_y, are the Markov chains that correspond to an application of the GHC algorithm to the discrete optimization problems D_y, y=1, 2, …, m for every n=1, 2, …, N(k) and for all states ω_1, ω_2, …, $\omega_{|\Omega_y|}$ as defined in Section 5.1.

Movement between discrete optimization problems is a stochastic process that satisfies the Markov property. To see this, define $\{\Psi(k)\}$, $\Psi(k) \in S$, k=1, 2, … to be the stochastic process where if $\Psi(k)=y$, then during outer loop iteration k, for all inner loop iterations n=1, 2, …, N(k) the SGHC algorithm is executing over solutions contained in the solution space of the discrete optimization problem $D_y=(\Omega_y, f_y)$. If the SGHC algorithm is executing over Ω_y at outer loop iteration k-1, then the probability that the SGHC algorithm is executing over Ω_q during outer loop iteration k is

$$T_{yq}(k) = h_{D_y D_q}(k, \rho(D_y, D_q))$$

independent of the discrete optimization problems the SGHC algorithm visited at outer loop iterations 1, 2, …, k-2 and independent of all preceding inner loop iterations. Therefore, the Markov property holds,

$$\Pr\{\Psi(k)=q \mid \Psi(k-1)=y,\ \Psi(k-2)=y_{k-2}, \ldots, \Psi(1)=y_1\} = \Pr\{\Psi(k)=q \mid \Psi(k-1)=y\} = T_{yq}(k).$$

Moreover, the Markov chain $\{\Psi(k)\}$ has transition matrix T(k), where $T_{yq}(k)$ is as defined above.

Chapter 6:

Simultaneous Generalized Hill Climbing Algorithm Analysis

Consider an application of the SGHC algorithm to a set of fundamentally related discrete optimization problems, S. This section presents sufficient conditions that guarantee that a SGHC algorithm will (as k approaches $+\infty$) be executing over the solution space of each discrete optimization problem $D_y \in S$ with probability $1/|S|$, where $|S|$ is the cardinality of S. This result implies that, as k approaches $+\infty$, each discrete optimization problem in $S=\{D_1, D_2, \ldots, D_m\}$ is being explored with equal probability.

This section develops sufficient conditions that place restrictions on the selection of the problem generation probability function $h_{D_y D_q}(k, \rho(D_y, D_q))$. Two sets of sufficient conditions are provided. The first set of sufficient conditions (Theorem 1) require selecting $h_{D_y D_q}(k, \rho(D_y, D_q))$ such that the associated Markov chain is stationary. A discrete time Markov chain is a *stationary Markov chain* if the probability of going from one state to another state is independent of the iteration at which the transition is being made (Isaacson and Madsen 1985). That is, let $\{X_n\}$ be a stationary Markov chain with state space $S=\{D_1, D_2, \ldots, D_m\}$, then for all states D_y and D_q,

$$\Pr\{X_n=D_y \mid X_{n-1}=D_q\} = \Pr\{X_{n+k}=D_y \mid X_{n+k-1}=D_q\},$$

for all k=-(n-1),-(n-2), …,-1, 0, 1, 2, … .

The *long run distribution (stationary probability distribution)* of a stationary Markov chain with corresponding transition matrix T is defined by $\pi = [\pi_1 \quad \pi_2 \quad \cdots \quad \pi_m]$, $\pi_i \geq 0$, for all i, i=1, 2, …, m, where $\pi=\pi T$ and $\sum_{i=1}^{m} \pi_i = 1$. Equivalently, the long run distribution of a stationary Markov chain is defined by $\pi = [\pi_1 \quad \pi_2 \quad \cdots \quad \pi_m]$, where

$$\pi_j = \lim_{n \to +\infty} T_{ij}^{(n)}.$$

The second set of sufficient conditions are presented in Theorem 2. Theorem 2 requires selecting $h_{D_y D_q}(k, \rho(D_y, D_q))$ such that the associated Markov chain is nonstationary. A discrete time Markov chain is a *nonstationary Markov chain* if the condition for stationary fails.

6.1 Stationary Markov Chain Sufficient Conditions

If the stationary Markov chain $\{\Psi(k)\}$ has a uniform long run distribution, then as k approaches infinity the SGHC algorithm is executing over the solution space of each discrete optimization problem in $S = \{D_1, D_2, \ldots, D_m\}$ with probability $1/m=1/|S|$. Theorem 1 provides sufficient conditions for the selection of the problem generation probability function $h_{D_y D_q}(k, \rho(D_y, D_q))$ that guarantee that the Markov chain $\{\Psi(k)\}$ has a uniform long run distribution. Therefore, when the sufficient conditions of Theorem 1 hold, the SGHC algorithm will (as k approaches $+\infty$) be in discrete optimization problem $D_y \in S$, y=1, 2, …, m with probability $1/|S|$.

To prove Theorem 1, the following definitions are needed. A subset, C, of the state space, S, is *closed* if $P_{ij}=0$, for all $i \in C$ and $j \notin C$. A Markov chain is *irreducible* if there exists no nonempty closed set other than S itself. If S has a proper closed subset, it is *reducible*. State

ω_i is said to have *period d* if $P_{ii}^{n}=0$ whenever n is not divisible by d and d is the greatest integer with this property. A state with period 1 is said to be *aperiodic* (Isaacson and Madsen 1985).

Theorem 1

Consider an application of the SGHC algorithm. Define $h_{D_yD_q}(k, \rho(D_y, D_q))= h_{D_yD_q}(\rho(D_y, D_q))= h_{D_qD_y}(\rho(D_y, D_q))$, for every k=1, 2, … . Consider the transition matrix T defined by

$$T_{yq} = h_{D_yD_q}(\rho(D_y, D_q)).$$

If the transition matrix T is irreducible and aperiodic, then the Markov chain $\{\Psi(k)\}$ has a uniform long run distribution. Moreover, the long run distribution of $\{\Psi(k)\}$ is

$$\pi = [1/|S| \; 1/|S| \; \dots \; 1/|S|].$$

Proof:

Note that the problem generation probability function $h_{D_yD_q}(k, \rho(D_y, D_q))$ is independent of k. Therefore, the Markov chain $\{\Psi(k)\}$ is stationary. Recall, that the long run distribution of a stationary Markov chain is defined by $[\pi_1 \quad \pi_2 \quad \cdots \quad \pi_m]$, $\pi_i \geq 0$, for all i=1, 2, …, m which satisfies

$$\pi = \pi T, \qquad (1)$$

and

$$\sum_{i=1}^{m} \pi_i = 1. \qquad (2)$$

Since $\sum_{i=1}^{m} T_{ij} = \sum_{j=1}^{m} T_{ij} = 1$, then (1) and (2) have the solution $\pi_1 = \pi_2 = \cdots = \pi_m = 1/m = 1/|S|$. Moreover, this solution is unique since T is aperiodic and irreducible (Issacson and Madsen 1985). Therefore, the Markov chain $\{\Psi(k)\}$ has a uniform long run distribution.

□

Theorem 1 shows that if the problem generation probability function is chosen such that the Markov chain {Ψ(k)} is stationary and the associated transition matrix is symmetric, then the Markov chain {Ψ(k)} has a uniform stationary distribution. The second set of sufficient conditions allow for the development of a nonstationary Markov chain.

6.2 Nonstationary Markov Chain Sufficient Conditions

For finite nonstationary Markov chains, the transition matrices T(k) that contain the probabilities of moving from state D_y to state D_q at outer loop iteration k, are functions of k. To define weak and strong ergodicity of nonstationary Markov chains, several definitions are needed. Define the *one norm* of a vector $\mathbf{f}=(f_1, f_2, \ldots, f_m)$ by

$$\| \mathbf{f} \| = \sum_{i=1}^{m} | f_i | .$$

Define the *infinity norm* of matrix T(k) by

$$\|T(k)\| = \max_{i} \sum_{j=1}^{m} | T_{ij} |$$

(Atkinson 1989). Let T(1), T(2), ... be the transition matrices for a nonstationary Markov chain with starting vector $f^{(0)}$. Define $f^{(j,k)}=f^{(0)}T(j+1)T(j+2) \ldots T(k)$.

A nonstationary Markov chain is *weakly ergodic* if, for all j,

$$\lim_{n \to +\infty} \sup_{f^{(0)} g^{(0)}} \|f^{(j,n)}-g^{(j,n)}\| = 0,$$

where $f^{(0)}$ and $g^{(0)}$ are starting vectors. A nonstationary Markov chain is *strongly ergodic* if there exists a vector $\mathbf{q} = (q_1, q_2, \ldots, q_m)$, with $\| \mathbf{q} \|=1$ and $q_i \geq 0$, for i=1, 2, …, m such that, for all j,

$$\lim_{n \to +\infty} \sup_{f^{(0)}} \|f^{(j,n)}-\mathbf{q}\| = 0,$$

where $f^{(0)}$ is a starting vector (Isaacson and Madsen 1985). The following result from Isaacson and Madsen (1985) is needed in the proof of Lemma 2.

Lemma 1

Let {T(k)} be a sequence of transition matrices corresponding to a nonstationary weakly ergodic Markov chain. If there exists a corresponding sequence of left eigenvectors {π (k)}, for {T(k)}, satisfying

$$\sum_{k=1}^{+\infty} \| \pi(k) - \pi(k+1) \| < {}^{+}\infty,$$

then the chain is strongly ergodic and for every j,

$$\lim_{n \to +\infty} \sup_{f^{(0)}} \|f^{(j,n)} - \pi\| = 0,$$

where

$$\lim_{k \to +\infty} \pi(k) = \pi.$$

Proof:

See Isaacson and Madsen (1985).

□

Lemma 2 states that if the problem generation probability function is selected such that the corresponding nonstationary Markov chain {Ψ(k)}, Ψ(k)$\in$ S, k=1, 2, … is weakly ergodic and the transition matrices T(k) are symmetric for every k, then the nonstationary Markov chain is strongly ergodic.

Lemma 2

Consider an application of the SGHC algorithm. Assume that the nonstationary Markov chain {Ψ(k)}, Ψ(k)$\in$ S, k=1, 2, … is weakly ergodic and that $h_{D_y D_q}$(k, ρ(D_y,

$D_q))= h_{D_q D_y}$ (k, ρ(D_q, D_y)), for every k. Then the nonstationary Markov chain {Ψ(k)} is strongly ergodic and for every j,

$$\lim_{n \to +\infty} \sup_{f^{(0)}} \|f^{(j,n)} - \pi\| = 0,$$

where π = [1/|S| 1/|S| ... 1/|S|].

Proof:

Let {T(k)} be the sequence of transition matrices corresponding to the nonstationary weakly ergodic Markov chain {Ψ(k)}, Ψ(k)∈S, k=1, 2, … . Let {π(k)} be the corresponding sequence of left eigenvectors. From Lemma 1, it is sufficient to show

$$\sum_{k=1}^{+\infty} \| \pi(k) - \pi(k+1) \| < {}^{+}\infty,$$

where

$$\lim_{k \to +\infty} \pi\,(k) = [1/|S|\ 1/|S|\ ...\ 1/|S|].$$

From Theorem 1, for every k, the Markov chains corresponding to the transition matrices T(k) have uniform long run distributions. Therefore,

$$\pi(k) = \pi(k+1) = [1/|S|\ 1/|S|\ \ldots\ 1/|S|],$$

for all k, k=1, 2, …, K, hence

$$\sum_{k=1}^{+\infty} \| \pi(k) - \pi(k+1) \| < {}^{+}\infty$$

and

$$\lim_{k \to +\infty} \pi\,(k) = [1/|S|\ 1/|S|\ ...\ 1/|S|].$$

□

Lemma 2 shows that if the Markov chain {Ψ(k)}, Ψ(k)∈S, k=1, 2, … is weakly ergodic and the associated transition matrices {T(k)} are symmetric, then the Markov chain {Ψ(k)} is strongly ergodic and the SGHC algorithm will (as k approaches +∞) be executing over the solution space of each discrete optimization problem contained in the set of fundamentally

related discrete optimization problems with equal probability. Theorem 2 shows that this result implies that for every $\varepsilon>0$, there exists an outer loop iteration such that for all future outer loop iterations, the SGHC algorithm is executing over the solution space of each discrete optimization problem in the set S with probability $1/|S| \pm \varepsilon$.

Theorem 2

Consider an application of the SGHC algorithm. Assume that the Markov chain $\{\Psi(k)\}$, $\Psi(k) \in S$, k=1, 2, … is weakly ergodic and that $h_{D_y D_q}(k, \rho(D_y, D_q)) = h_{D_q D_y}(k, \rho(D_q, D_y))$, for every k=1, 2, … and for every q, y=1, 2, …, m (i.e., the corresponding transition matrix is symmetric). Then for every $\varepsilon>0$, there exists a $k(\varepsilon) \in Z^+$, such that

$$1/|S| - \varepsilon \le \Pr\{\Psi(k)=y\} \le 1/|S| + \varepsilon,$$

for all outer loop iterations $k \ge k(\varepsilon)$ and for every $D_y \in S$, y=1, 2, …, m.

Proof:

From Lemma 2,

$$\lim_{n \to +\infty} \sup_{f^{(0)}} \|f^{(1,n)} - \pi\| = 0,$$

where $\pi = [1/|S| \; 1/|S| \; ... \; 1/|S|]$. Therefore, for every $\varepsilon>0$, there exists a $k(\varepsilon) \in Z^+$, such that for all outer loop iterations $k \ge k(\varepsilon)$,

$$\sup_{f^{(0)}} \|f^{(1,k)} - \pi\| < \varepsilon.$$

This means that for every initial solution vector $f^{(0)}$ and for all outer loop iterations $k \ge k(\varepsilon)$,

$$\|f^{(1,k)} - \pi\| < \varepsilon.$$

Therefore, independent of the initial discrete optimization problem, for every $\varepsilon>0$ and for every discrete optimization $D_y \in S$, y=1, 2, …, m, there exists an $k(\varepsilon) \in Z^+$ such that, for all outer loop iterations $k \ge k(\varepsilon)$,

$$1/|S| - \varepsilon \leq \Pr\{\Psi(k)=y\} \leq 1/|S| + \varepsilon.$$

□

6.3 Conditions for Weak Ergodicity

Consider an application of the SGHC algorithm. It is beneficial to select the problem generation probability function $h_{D_y D_q}(k, \rho(D_y, D_q))$ such that the corresponding Markov chain $\{\Psi(k)\}$ is weakly ergodic. For example, Theorem 2 of Section 6.2 provides sufficient conditions that guarantee (as k approaches $+\infty$) the SGHC algorithm will be executing over the solution space of each discrete optimization problem in the set S with equal probability. These sufficient conditions require that $h_{D_y D_q}(k, \rho(D_y, D_q))$ be defined such that the associated Markov chain $\{\Psi(k)\}$, $\Psi(k) \in S$, k=1, 2, … is weakly ergodic. Moreover, for an application of a SGHC algorithm, if the Markov chain $\{\Psi(k)\}$, $\Psi(k) \in S$, k=1, 2, … is weakly ergodic, then for sufficiently large k, the probability that the SGHC algorithm is executing over the solution space of a particular discrete optimization problem during outer loop iteration k is independent of the initial discrete optimization problem. This guarantees that the long-term performance of a SGHC algorithm is independent of the initial discrete optimization problem. This section provides sufficient conditions for the selection of the problem generation probability function $h_{D_y D_q}(k, \rho(D_y, D_q))$ that guarantee that the corresponding Markov chain $\{\Psi(k)\}$ is weakly ergodic.

Examples 6-1, 6-2 and 6-3 illustrate nonstationary Markov chains that are weakly ergodic.

Example 6-1

Consider the nonstationary Markov chain with transition matrices,

$$T(k) = \begin{bmatrix} 1-(1/(k+1)) & 1/2(k+1) & 1/2(k+1) \\ 1/2(k+1) & 1-(1/(k+1)) & 1/2(k+1) \\ 1/2(k+1) & 1/2(k+1) & 1-(1/(k+1)) \end{bmatrix}, \quad k=1, 2, \ldots .$$

Table 6-1 provides $f^{(1,k)} = f^{(0)} T(1)T(2) \ldots T(k)$ for the initial vectors $f^{(0)}$= [1 0 0], [0 1 0] and [0 0 1] and for k=10,000 and k=20,000.

Table 6-1: Weakly Ergodic Example One

Initial Vector $f^{(0)}$	$f^{(1, 10,000)}$	$f^{(1, 20,000)}$
[1 0 0]	[.3333 .3333 .3333]	[.3333 .3333 .3333]
[0 1 0]	[.3333 .3333 .3333]	[.3333 .3333 .3333]
[0 0 1]	[.3333 .3333 .3333]	[.3333 .3333 .3333]

Example 6-2

Consider the nonstationary Markov chain with transition matrices,

$$T(k) = \begin{bmatrix} 1-(1/(k+1)) & 1/2(k+1) & 1/2(k+1) \\ 0 & 1-1/(k+1) & 1/(k+1) \\ 1/(k+1) & 0 & 1-1/(k+1) \end{bmatrix}, \quad k=1, 2, \ldots .$$

Table 6-2 provides $f^{(1,k)} = f^{(0)} T(1)T(2) \ldots T(k)$ for the initial vectors $f^{(0)}$= [1 0 0], [0 1 0] and [0 0 1] and for k=10,000 and k=20,000.

Table 6-2: Weakly Ergodic Example Two

Initial Vector $f^{(0)}$	$f^{(1, 10,000)}$	$f^{(1, 20,000)}$
[1 0 0]	[.4 .2 .4]	[.4 .2 .4]
[0 1 0]	[.4 .2 .4]	[.4 .2 .4]
[0 0 1]	[.4 .2 .4]	[.4 .2 .4]

Example 6-3

Consider the nonstationary Markov chain with transition matrices,

$$T(k) = \begin{bmatrix} 1-(1/((k+1)^2)) & 1/(2(k+1)^2) & 1/(2(k+1)^2) \\ 1/2(k+1) & 1-(1/(k+1)) & 1/2(k+1) \\ 1/2(k+1) & 1/2(k+1) & 1-(1/(k+1)) \end{bmatrix}, \quad k=1, 2, \ldots .$$

Table 6-3 provides $f^{(1,k)} = f^{(0)} T(1)T(2) \ldots T(k)$ for the initial vectors $f^{(0)}$ = [1 0 0], [0 1 0] and [0 0 1] and for k=10,000, k=20,000 and k=100,000.

Table 6-3: Weakly Ergodic Example Three

Initial Vector $f^{(0)}$	$f^{(1, 10,000)}$	$f^{(1, 20,000)}$	$f^{(1, 100,000)}$
[1 0 0]	[.9918 .0041 .0041]	[.9904 .0048 .0048]	[.9957 .0022 .0022]
[0 1 0]	[.9817 .0092 .0092]	[.9870 .0065 .0065]	[.9942 .0029 .0029]
[0 0 1]	[.9817 .0092 .0092]	[.9870 .0065 .0065]	[.9942 .0029 .0029]

Examples 6-4 and 6-5 illustrate nonstationary Markov chains that are not weakly ergodic.

Example 6-4

Consider the nonstationary Markov chain with transition matrices,

$$T(k) = \begin{bmatrix} 1-(1/(k+1)^2) & 1/2(k+1)^2 & 1/2(k+1)^2 \\ 1/2(k+1)^2 & 1-(1/(k+1)^2) & 1/2(k+1)^2 \\ 1/2(k+1)^2 & 1/2(k+1)^2 & 1-(1/(k+1)^2) \end{bmatrix}, \quad k=1, 2, \ldots .$$

Table 6-4 provides $f^{(1,k)} = f^{(0)} T(1)T(2) \ldots T(k)$ for the initial vectors $f^{(0)}$= [1 0 0], [0 1 0] and [0 0 1] and for k=10,000, k=20,000 and k=100,000.

Table 6-4: Not Weakly Ergodic Example Four

Initial Vector $f^{(0)}$	$f^{(1, 10,000)}$	$f^{(1, 20,000)}$	$f^{(1, 100,000)}$
[1 0 0]	[.5567 .4429 .0004]	[.5582 .2209 .2209]	[.5582 .2209 .2209]
[0 1 0]	[.4429 .5567 .0004]	[.2209 .5582 .2209]	[.2209 .5582 .2209]
[0 0 1]	[.4998 .4998 .0004]	[.2209 .2209 .5582]	[.2209 .2209 .5582]

Example 6-5

Consider the nonstationary Markov chain with transition matrices,

$$T(k) = \begin{bmatrix} 1-(1/(k+1)^2) & 1/2(k+1)^2 & 1/2(k+1)^2 \\ 1/2(k+1)^2 & 1-(1/(k+1)^2) & 1/2(k+1)^2 \\ 1/2(k+1) & 1/2(k+1) & 1-(1/(k+1)) \end{bmatrix}, \quad k=1, 2, \ldots .$$

Table 6-5 provides $f^{(1,k)} = f^{(0)} T(1)T(2) \ldots T(k)$ for the initial vectors $f^{(0)}$= [1 0 0], [0 1 0] and [0 0 1] and for k=10,000 and k=20,000.

Table 6-5: Not Weakly Ergodic Example Five

Initial Vector $f^{(0)}$	$f^{(1, 10,000)}$	$f^{(1, 20,000)}$
[1 0 0]	[1 0 0]	[1 0 0]
[0 1 0]	[0 1 0]	[0 1 0]
[0 0 1]	[0 0 1]	[0 0 1]

Examples 6-1 through 6-5 illustrate how transition matrices that correspond to weakly ergodic nonstationary Markov chains can look similar to transition matrices that correspond to non-weakly ergodic nonstationary Markov chains. To determine how to differentiate between transition matrices that correspond to weakly ergodic nonstationary Markov chains and transition matrices that correspond to non-weakly ergodic nonstationary Markov chains the following definitions are needed.

The *ergodic coefficient* of T(k), denoted by $\alpha(T(k))$, is defined by

$$\alpha(T(k)) = 1 - \sup_{i,l} \sum_{j=1}^{m} \left[T_{ij}(k) - T_{lj}(k)\right]^{+},$$

where $\left[T(k)_{ij} - T(k)_{lj}\right]^{+} = \max(0, T(k)_{ij} - T(k)_{lj})$. The *delta coefficient* of T(k) is $\delta(T(k))=1-\alpha(T(k))$ (Isaacson and Madsen 1985). Figure 6.1 contains the ergodic and delta coefficients of the transition matrices T(k) in Examples 6-1 through 6-5.

Example	α	δ
6-1	$3/(2(k+1))$	$1-3/(2(k+1))$
6-2	$1/(k+1)$	$1-1/(k+1)$
6-3	$3/(2(k+1))$	$1-3/(2(k+1))$
6-4	$3/(2(k+1)^2)$	$1-3/(2(k+1)^2)$
6-5	$3/(2(k+1)^2)$	$1-3/(2(k+1)^2)$

Figure 6.1: Ergodic Coefficients

Theorem 3 and Theorem 4 (Isaacson and Madsen 1985) present necessary and sufficient conditions that relate weak ergodicity to the ergodic coefficient.

Theorem 3

Let $\{X_n\}$ be a nonstationary Markov chain with transition matrices T(k), k=1, 2, … . The chain is weakly ergodic if and only if there exists a subdivision of T(1)T(2)T(3) … into blocks of matrices [T(1)T(2)T(3) … $T(n_1)$] … [$T(n_1+1)T(n_1+2)$ … $T(n_2)$] … such that

$$\sum_{j=0}^{+\infty} \alpha(T(n_j+1)T(n_j+2)\ldots T(n_{j+1})) = +\infty,$$

where $n_0=0$.

Proof:

See Isaacson and Madsen (1985).

□

Theorem 4

A nonstationary Markov chain is weakly ergodic if and only if, for all $j \in Z^+$,

$$\lim_{k \to +\infty} \delta(T(j)T(j+1) \ldots T(k)) = 0.$$

Proof:

See Isaacson and Madsen (1985).

□

Theorem 5 states sufficient conditions based on the selection of the problem generation probability function $h_{D_yD_q}(k, \rho(D_y, D_q))$ that guarantee that the Markov chain $\{\Psi(k)\}$ is weakly ergodic.

Theorem 5

Consider an application of the SGHC algorithm. Suppose that $h_{D_yD_q}(k, \rho(D_y, D_q)) = h_{D_qD_y}(k, \rho(D_q, D_y))$, for every k=1, 2, …, K and for every y, q=1,2, …, m (i.e., the corresponding transition matrix T(k) is symmetric). Consider the transition matrices T(k) defined by

$$T_{yq}(k) = h_{D_yD_q}(k, \rho(D_y, D_q)),$$

corresponding to stochastic process $\{\Psi(k)\}$, $\Psi(k) \in S$, k=1, 2, … . If $\delta(T(k)) \le 1 - 1/(k+1)$, for every k, then the Markov chain $\{\Psi(k)\}$ is weakly ergodic.

Proof:

From Theorem 3, it is sufficient to show that

$$\sum_{k=1}^{+\infty} \alpha(T(k)) = +\infty,$$

which follows from

$$\begin{aligned}\sum_{k=1}^{+\infty} \alpha(T(k)) &= \sum_{k=1}^{+\infty} (1 - \delta(T(k))) \\ &\ge \sum_{k=1}^{+\infty} 1/(k+1) \\ &= +\infty.\end{aligned}$$

□

Theorem 6 provides an exact problem generation probability function $h_{D_yD_q}(k, \rho(D_y, D_q))$ that guarantees that the Markov chain $\{\Psi(k)\}$ is weakly ergodic. Note that the problem generation probability function given in Theorem 6 does not satisfy the sufficient conditions of Theorem 5.

Theorem 6

Consider an application of the SGHC algorithm to a set of discrete optimization problems S, where |S|>2. If $h_{D_y D_y}(k, \rho(D_y, D_y))=1-1/(k+1)$ for every k=1, 2, …, K and for every y=1, 2, …, m and $h_{D_y D_q}(k, \rho(D_y, D_q))=1/((|S|-1)(k+1))$, for every k=1, 2, …, K and for every q, y=1, 2, …, m, where q≠y, then the Markov chain {Ψ(k)} is weakly ergodic.

Proof:

Note that $\delta(T(k))=1-[(|S|-2)/((|S|-1)(k+1))]$. From Theorem 3, it is sufficient to show that

$$\sum_{k=1}^{+\infty} \alpha(T(k)) = +\infty,$$

which follows from

$$\sum_{k=1}^{+\infty} \alpha(T(k)) = \sum_{k=1}^{+\infty} (1-\delta(T(k)))$$

$$=(|S|-2)/(|S|-1)\sum_{k=1}^{+\infty} 1/(k+1)$$

$$= +\infty.$$

□

Theorem 6 provides an exact problem generation probability function $h_{D_y D_q}(k, \rho(D_y, D_q))$ that guarantees that the Markov chain {Ψ(k)} is weakly ergodic, where the problem generation probability function is only a function of the outer loop iteration k=1, 2, …, K. Theorem 7 provides an exact problem generation probability function $h_{D_y D_q}(k, \rho(D_y, D_q))$ that guarantees that the Markov chain {Ψ(k)} is weakly ergodic, where the problem generation probability function is a function of both the outer loop iteration k=1, 2, …, K and the detachment metric $\rho(D_y, D_q)$.

Theorem 7

Consider an application of the SGHC algorithm to a set of discrete optimization problems S where $\rho(D_y, D_q)\neq 0$, for all $y\neq q$. If $h_{D_yD_y}(k, \rho(D_y, D_y))=1-1/(k+1)$ for every k=1, 2, …, K and for every y=1, 2, …, m and

$$h_{D_yD_q}(k, \rho(D_y, D_q)) = [1/(\rho(D_y, D_q)(k+1))] / [\sum_{\substack{i=1 \\ i\neq y}}^{m} (1/\rho(D_y, D_i))],$$

for every k=1, 2, …, K and for every q,y=1, 2, …, m, $y\neq q$, then the Markov chain $\{\Psi(k)\}$ is weakly ergodic.

Proof:

Let

$$\varepsilon=\min\{[1/(\rho(D_y, D_q)(k+1))] / [\sum_{\substack{i=1 \\ i\neq y}}^{m} (1/\rho(D_y, D_i))], y\neq q, y,q=1, 2, \ldots, m\}.$$

Note that $0<\varepsilon<+\infty$, since $\rho(D_y, D_q)\neq 0$, for all $y\neq q$. Then,

$$\delta(T(k)) \leq 1-1/(k+1)+(1/(k+1))\sum_{\substack{q=1 \\ q\neq y}}^{m}\left(\frac{1/\rho(D_y, D_q)}{\sum_{\substack{i=1 \\ i\neq y}}^{m} 1/\rho(D_y, D_i)} - \varepsilon\right),$$

for some y=1, 2, …, m.

$$= 1-\varepsilon(|S|-1)/(k+1),$$

where $\varepsilon>0$.

From Theorem 3, it is sufficient to show that

$$\sum_{k=1}^{+\infty}\alpha(T(k)) = +\infty,$$

which follows from

$$\sum_{k=1}^{+\infty} \alpha(T(k)) = \sum_{k=1}^{+\infty} (1 - \delta(T(k)))$$

$$\geq \varepsilon(|S|-1) \sum_{k=1}^{+\infty} 1/(k+1)$$

$$= +\infty.$$

□

Chapter 7:

Performance of Simultaneous Generalized Hill Climbing Algorithms

This chapter studies the performance of applications of the SGHC algorithm where the problem generation probability function meets the following conditions (referred to as Criteria A).

Criteria A:

(i) $h_{D_y D_q}(k, \rho(D_y, D_q)) = h_{D_q D_y}(k, \rho(D_q, D_y))$,

(ii) $\lim_{k \to +\infty} h_{D_y D_y}(k, \rho(D_y, D_y)) = 1$,

(iii) $h_{D_y D_y}(k, \rho(D_y, D_y))$ is monotonically increasing

(iv) the Markov chain $\{\Psi(k)\}$ is weakly ergodic,

for every q, y=1, 2, …, m and for every k=1, 2, …, K.

Note that, by Lemma 2, if a SGHC algorithm satisfies Criteria A, then the Markov chain $\{\Psi(k)\}$ is strongly ergodic.

Condition (i) in Criteria A requires that the transition matrices {T(k)} corresponding to the Markov chain {Ψ(k)} are symmetric. Since, the detachment metric is symmetric (i.e., $\rho(D_y, D_q) = \rho(D_q, D_y)$) this condition can be easily satisfied. Condition (ii) in Criteria A requires that, as the outer loop iterations k=1, 2, … approach +∞, the probability that the SGHC algorithm will stay in the current discrete optimization problem is approaching one. Therefore, the transition matrices {T(k)} corresponding to the Markov chain {Ψ(k)} must be such that the diagonal elements approach one as k approaches +∞. Condition (iii) in Criteria A requires that these diagonal elements are monotonically increasing. This condition implies that as k increases so does the probability that the SGHC algorithm will stay in a given discrete optimization problem.

Condition (iv) in Criteria A requires that the Markov chain {Ψ(k)} is weakly ergodic. This is the most difficult condition to verify when specifying the problem generation probability function for a SGHC algorithm. Section 6.3 focused on this difficulty. Recall, the Markov chain {Ψ(k)} is weakly ergodic if the problem generation probability function satisfies the sufficient conditions of Theorem 5, Theorem 6 or Theorem 7. Moreover, if the problem generation probability function for a SGHC algorithm satisfies the sufficient conditions of Theorem 6 or Theorem 7, then the SGHC algorithm satisfies Criteria A.

7.1 The Simultaneous Generalized Hill Climbing Algorithm Visits Each Discrete Optimization Problem Infinitely Often

This section shows that a SGHC algorithm that satisfies Criteria A visits each discrete optimization problem in the set of fundamentally related discrete optimization problems infinitely often.

Theorem 2 shows that for all ε>0, there exists an outer loop iteration k(ε) such that the SGHC algorithm is executing over the solution space of each discrete optimization problem in the set S with probability $1/|S| \pm \varepsilon$, for all outer loop iterations $k \geq k(\varepsilon)$. Lemma 3 is an extension

of Theorem 2. Lemma 3 shows that for all ε>0 and for every positive integer $\bar{k}$ there exists an outer loop iteration k(ε)≥$\bar{k}$, such that for all outer loop iterations k≥k(ε), the SGHC algorithm will be in each discrete optimization problem in the set S with probability 1/|S| ± ε.

Lemma 3

Consider an application of the SGHC algorithm where the problem generation probability function satisfies Criteria A. Then for all ε>0, where 1/|S|>ε, and for every $\bar{k} \in Z^+$, there exists an outer loop iteration k(ε)≥$\bar{k}$, such that for every outer loop iteration k≥k(ε),

$$1/|S| - \varepsilon \leq \Pr\{\Psi(k)=y\} \leq 1/|S| + \varepsilon$$

for every $D_y \in S$, y = 1, 2, …, m.

Proof:

Since the Markov chain {Ψ(k)} is strongly ergodic, by Lemma 2 for every j,

$$\lim_{n \to +\infty} \sup_{f^{(0)}} \|f^{(j,n)} - \pi\| = 0,$$

where,

$$\pi = [1/|S| \; 1/|S| \; \ldots \; 1/|S|].$$

In particular,

$$\lim_{n \to +\infty} \sup_{f^{(0)}} \|f^{(\bar{k},n)} - \pi\| = 0.$$

Therefore, for all ε>0, there exists k(ε)≥$\bar{k}$ such that for every outer loop iteration k≥k(ε),

$$\sup_{f^{(0)}} \|f^{(\bar{k},k)} - \pi\| < \varepsilon.$$

Hence,

$$1/|S| - \varepsilon \leq \Pr\{\Psi(k)=y\} \leq 1/|S| + \varepsilon$$

for every $D_y \in S$, y=1, 2, …, m and for every $k \geq k(\varepsilon) \geq \bar{k}$.

□

Lemma 4 is needed in the proof of Theorem 8. Lemma 4 shows that for every $D_y \in S$, for all $\varepsilon>0$ and for every positive integer $\bar{k}$, there exist an infinite sequence of outer loop iterations $k_1(\varepsilon)$, $k_2(\varepsilon)$, $k_3(\varepsilon)$ …, where $\bar{k} < k_1(\varepsilon) < k_2(\varepsilon) < k_3(\varepsilon)$ … such that the SGHC algorithm is executing over the solution space of D_y during outer loop iterations $k_1(\varepsilon)$, $k_2(\varepsilon)$, $k_3(\varepsilon)$, … with probability $1/|S| \pm \varepsilon$.

Lemma 4

Consider an application of the SGHC algorithm where the problem generation probability function satisfies Criteria A. Then for all $\varepsilon>0$, for every $\bar{k} \in Z^+$, and for every discrete optimization problem $D_y \in S$, y=1, 2, …, m, there exists a sequence of outer loop iterations $k_1(\varepsilon)$, $k_2(\varepsilon)$, $k_3(\varepsilon)$, … such that

$$\bar{k} < k_1(\varepsilon) < k_2(\varepsilon) < k_3(\varepsilon) \ldots$$

and

$$1/|S| - \varepsilon \leq Pr\{\Psi(k_i(\varepsilon))=y\} \leq 1/|S| + \varepsilon$$

for every i=1, 2, … .

Proof:

The proof is by induction.

Base case:

From Lemma 3, there exists $k(\varepsilon) \geq \bar{k}$ such that

$$1/|S| - \varepsilon \leq Pr\{\Psi(k)=y\} \leq 1/|S| + \varepsilon$$

for every $k \geq k(\varepsilon)$. Therefore, there exists an outer loop iteration $k_1(\varepsilon) > \bar{k}$

$$1/|S| - \varepsilon \leq Pr\{\Psi(k_1(\varepsilon))=y\} \leq 1/|S| + \varepsilon.$$

Induction Step:

Assume there exists outer loop iterations $k_1(\varepsilon)$, $k_2(\varepsilon)$, …, $k_j(\varepsilon)$ such that

$$\bar{k} < k_1(\varepsilon) < k_2(\varepsilon) < \ldots < k_j(\varepsilon)$$

and

$$1/|S| - \varepsilon \le \Pr\{\Psi(k_i(\varepsilon))=y\} \le 1/|S| + \varepsilon$$

for all i=1, 2, …, j.

From Lemma 3, there exists $k(\varepsilon) \ge k_j(\varepsilon)$ such that

$$1/|S| - \varepsilon \le \Pr\{\Psi(k)=y\} \le 1/|S| + \varepsilon$$

for every $k \ge k(\varepsilon)$. Therefore, there exists an outer loop iteration $k_{j+1}(\varepsilon) > k_i(\varepsilon)$

$$\bar{k} < k_1(\varepsilon) < k_2(\varepsilon) < \ldots < k_{j+1}(\varepsilon)$$

and

$$1/|S| - \varepsilon \le \Pr\{\Psi(k_i(\varepsilon))=y\} \le 1/|S| + \varepsilon$$

for all i=1, 2, …, j+1.

□

Theorem 8 guarantees that, for every $\bar{k} \in Z^+$ and for every discrete optimization problem D_y, y=1, 2, …, m, there exists an outer loop iteration k, where $k > \bar{k}$ and the SGHC will be executing over the solution space of discrete optimization problem D_y during outer loop iteration k.

Theorem 8

Consider an application of the SGHC algorithm where the problem generation probability function satisfies Criteria A. Then for every $\bar{k} \in Z^+$, and for every discrete optimization problem $D_y \in S$, y=1, 2, …, m, there exists an outer loop iteration $k > \bar{k}$ such that $\Psi(k)=y$.

Proof:

From Lemma 4, there exists a sequence of outer loop iterations $k_1(\varepsilon)$, $k_2(\varepsilon)$, $k_3(\varepsilon)$, … such that

$$\bar{k} < k_1(\varepsilon) < k_2(\varepsilon) < k_3(\varepsilon) \ldots$$

and

$$1/|S| - \varepsilon \leq \Pr\{\Psi(k_i(\varepsilon))=y\} \leq 1/|S| + \varepsilon$$

for every i=1, 2, … .

Therefore,

$$\Pr\{\Psi(k)\neq y, \text{ for all } k>\bar{k}\} \leq \Pr\{\Psi(k_i(\varepsilon))\neq y, \text{ for all } i=2, \ldots\}$$

$$= \lim_{n \to +\infty} (1-1/|S| \pm \varepsilon)^n = 0.$$

□

7.2 The Expected Number of Iterations the Simultaneous Generalized Hill Climbing Algorithm Spends in Each Discrete Optimization Problem

This section investigates the expected number of outer loop iterations that a SGHC algorithm that satisfies Criteria A will execute over the solution space of a particular discrete optimization problem from the set of fundamentally related discrete optimization problems S. Moreover, this section develops a lower bound for the probability that a SGHC algorithm that satisfies Criteria A will, for a given number of outer loop iterations, continue to execute over the solution space of a particular discrete optimization problem.

Lemma 5 shows that the expected number of outer loop iterations that the SGHC algorithm is executing over the solution space of a particular discrete optimization problem has a lower bound that is a function of the problem generation probability function.

Lemma 5

Consider an application of the SGHC algorithm where the problem generation probability function satisfies Criteria A. Let $X_k(D_y)$ be the number of consecutive outer loop iterations, starting at k, such that the SGHC algorithm is in D_y. Then

$$E[X_k(D_y)|\ \Psi(k-1)=y] \geq \frac{h_{D_yD_y}(k, \rho(D_y, D_y))}{(1-h_{D_yD_y}(k, \rho(D_y, D_y)))}.$$

Proof:

$$E[X_k(D_y)|\ \Psi(k-1)=y] = \sum_{n=1}^{+\infty} nPr\{X_k= n\}$$

$$= \sum_{n=1}^{+\infty} n(1- h_{D_yD_y}((k+n), \rho(D_y, D_y))) \prod_{i=0}^{n-1} h_{D_yD_y}(k+i, \rho(D_y, D_y))$$

$$= \sum_{n=1}^{+\infty} \prod_{i=0}^{n-1} h_{D_yD_y}(k+i, \rho(D_y, D_y))$$

Since $h_{D_yD_y}(k, \rho(D_y, D_y))$ is monotonically increasing and $0< h_{D_yD_y}(k, \rho(D_y, D_y))<1$,

$$E[X_k(D_y)|\ \Psi(k-1)=y] \geq \sum_{n=1}^{+\infty} (h_{D_yD_y}(k, \rho(D_y, D_y)))^n$$

$$= \frac{h_{D_yD_y}(k, \rho(D_y, D_y))}{(1-h_{D_yD_y}(k, \rho(D_y, D_y)))}.$$

□

Consider a particular discrete optimization problem, $D_y \in S$. Theorem 9 shows that for every positive integer M, there exists $k \in Z^+$, such that if the SGHC algorithm is executing over the solution space of D_y during outer loop iteration k-1, then the expected number of outer loop iterations that the SGHC algorithm will continue to execute over the solution space of D_y is at least M.

Theorem 9

Consider an application of the SGHC algorithm where the problem generation probability function satisfies Criteria A. Then for every discrete optimization problem $D_y \in S$ and for every $M \in Z^+$, there exists a outer loop iteration k(M) such that for all outer loop iterations $k \geq k(M)$

$$E[X_k(D_y) \mid \Psi(k-1)=y] \geq M.$$

Proof:

Let $D_y \in S$ and $M \in Z^+$ be given. Since $\lim_{k \to \infty} h_{D_y D_y}(k, \rho(D_y, D_y))=1$, for every $\frac{1}{M+1}>0$, then there exists a outer loop iteration k(M) such that for every $k \geq k(M)$

$$\| 1- h_{D_y D_y}(k, \rho(D_y, D_y)) \| \leq \frac{1}{M+1}.$$

Therefore, from Lemma 5, for every $k \geq k(M)$,

$$E[X_k(D_y) \mid \Psi(k-1)=y] = \sum_{n=1}^{+\infty} nPr\{X_k= n\}$$

$$\geq \frac{h_{D_y D_y}(k, \rho(D_y, D_y))}{(1-h_{D_y D_y}(k, \rho(D_y, D_y)))}$$

$$\geq \frac{1}{\| 1-h_{D_y D_y}(k, \rho(D_y, D_y)) \|} -1$$

$$\geq M.$$

□

Theorem 10 shows that there is a lower bound for the probability that a SGHC algorithm that satisfies Criteria A will continue to execute over the solution space of a particular discrete optimization problem from the set of fundamentally related discrete optimization problems.

Theorem 10

Consider an application of the SGHC algorithm where the problem probability function satisfies Criteria A, then for every outer loop iteration k

$$P\{X_k(D_y)>M \mid \Psi(k-1)=y\} \geq h_{D_yD_y}(k, \rho(D_y, D_y))^M.$$

Proof:

$$P\{X_k(D_y)>M \mid \Psi(k-1)=y\}=1- P\{X_k(D_y)\leq M \mid \Psi(k-1)=y\}$$

$$=1-\sum_{n=0}^{M} P\{X_k(D_y)=n \mid \Psi(k-1)=y\}$$

$$= h_{D_yD_y}(k, \rho(D_y, D_y)))-\sum_{n=1}^{M} (1- h_{D_yD_y}(k+n, \rho(D_y, D_y))) \prod_{i=0}^{n-1} h_{D_yD_y}(k+i, \rho(D_y, D_y))$$

$$=\prod_{i=0}^{M} h_{D_yD_y}(k+i, \rho(D_y, D_y)).$$

Since $h_{D_yD_y}(k, \rho(D_y, D_y))$ is monotonically increasing in k,

$$\geq (h_{D_yD_y}(k, \rho(D_y, D_y)))^M.$$

□

7.3 Convergence of Simultaneous Generalized Hill Climbing Algorithms

This section develops sufficient conditions that guarantee that a SGHC algorithm that satisfies Criteria A will visit the globally optimal solution over the set of fundamentally related discrete optimization problems. Recall, for all inner loop iterations the SGHC algorithm is executing a GHC algorithm over the solution space of a particular discrete optimization problem from the set of fundamentally related discrete optimization problems. This GHC algorithm will be referred to as the underlying GHC algorithm with hill climbing random variable $\overline{R}_k$. Define $\{Q_{\overline{n}}^{\overline{k}}(D_y)\}$, $\overline{k}$ =1, 2, …, $\overline{n}$ =1, 2, …, $\overline{N}(\overline{k})$, y=1, 2, …, m to be the Markov chain corresponding to the underlying GHC algorithm executing over the solution space of D_y, y=1, 2, …, m (see Sections 5.1 and 5.2).

Lemma 6 is needed for the proof of Theorem 11. Lemma 6 shows that for every discrete optimization problem D_y in the set of fundamentally related discrete optimization problems there exists an infinite sequence of outer loop iterations such that the SGHC algorithm is executing over the solution space of D_y during these outer loop iterations.

Lemma 6

Consider an application of the SGHC algorithm where the problem generation probability function satisfies Criteria A. For every $\bar{k} \in Z^+$ there exists, k_1, k_2, … such that

$$\bar{k} < k_1 < k_2 < k_3 \ldots$$

and

$$\Psi(k_i) = D_y,$$

for all y=1, 2, …, m and for all outer loop iterations k_i, for all i=1, 2, … .

Proof:

The proof is by induction.

Base Case: From Theorem 8, there exists $k_1 > \bar{k}$ such that

$$\Psi(k_1) = D_y.$$

Induction Step: Assume there exist k_1, k_2, …, k_j, such that

$$\bar{k} < k_1 < k_2 < \ldots < k_j,$$

and

$$\Psi(k_i) = D_y,$$

for all i=1, 2, …, j.

From Theorem 8, there exists $k_{j+1} > k_j$ such that

$$\bar{k} < k_1 < k_2 < \ldots < k_{j+1},$$

and

$$\Psi(k_i) = D_y,$$

for all i=1, 2, …, j+1.

□

Theorem 11 is needed in the proof of Theorem 12. Theorem 11 shows that for every discrete optimization problem $D_y \in S$, there is an outer loop iteration k such that the SGHC algorithm is executing over the solution space of D_y during outer loop iteration k and continues executing over the solution space of D_y for more than M outer loop iterations.

Theorem 11

Consider an application of the SGHC algorithm where the problem generation probability function satisfies Criteria A. For every discrete optimization problem D_y, y=1, 2, …, m and for every $M \in Z^+$, there exists an outer loop iteration $k=k(D_y, M)$, such that for outer loop iterations $k' = k, k+1, \ldots, k+M$,

$$\Psi(k') = y.$$

Proof:

The proof is by contradiction. Assume that there does not exist a k such that

$$\Psi(k') = y,$$

$k' = k, k+1, \ldots, k+M$. From Lemma 6, there exists $k_1, k_2, \ldots$ such that

$$k_1-1 < k_2-1 < k_3-1 \ldots$$

and

$$\Psi(k_i-1) = D_y,$$

for all i=1, 2, … .

From Theorem 10,

$$\Pr\{X_{k_i}(D_y)>M\}=\Pr\{X_{k_i}(D_y)>M \mid \Psi(k_i-1)=y\} \geq h_{D_yD_y}(k_i, \rho(D_y, D_y))^M.$$

Therefore,

$$\Pr\{X_{k_i}(D_y)\leq M\} \leq 1-(h_{D_yD_y}(k_i, \rho(D_y, D_y)))^M.$$

Note that,

$$\Pr\{X_{k_i}(D_y) > M, \text{ for some } k_i, i=1, 2, \ldots .\}$$

$$\leq \Pr\{X_k(D_y) > M, \text{ for some } k=1, 2, \ldots .\}.$$

Therefore,

$$1 - \Pr\{X_{k_i}(D_y) > M, \text{ for some } k_i, i=1, 2, \ldots .\}$$

$$\geq 1 - \Pr\{X_k(D_y) > M, \text{ for some } k=1, 2, \ldots .\}.$$

Hence,

$$\Pr\{X_k(D_y) \leq M \text{ , for all } k=1, 2, \ldots .\}$$

$$\leq \Pr\{X_{k_i}(D_y) \leq M, \text{ for all } i=1, 2, \ldots .\}$$

$$\leq \prod_{i=1}^{+\infty} (1-(h_{D_yD_y}(k_i, \rho(D_y, D_y)))^M) = 0$$

since,

$$\lim_{k \to +\infty} h_{D_yD_y}(k, \rho(D_y, D_q)) = 1.$$

This is a contradiction.

□

Lemma 7, Lemma 8 and Theorem 12 consider an application of the SGHC algorithm that satisfies Criteria A where for all outer loop iterations k, if a new discrete optimization problem is generated, the hill climbing random variable, the generation probability function and the inner loop bounds are reset (i.e., $R_k = \overline{R}_1$, $g_{ij}(k) = g_{ij}(1)$, $N(k) = \overline{N}(1)$).

Lemma 7 and Lemma 8 are needed for the proof of Theorem 12. Lemma 7 shows that if a new discrete optimization problem D_y is generated at outer loop iteration k and the SGHC algorithm executes over the solution space of D_y for outer loop iterations k, k+1, …, k+M, then during outer loop iterations k, k+1, …, k+M the SGHC algorithm can be modeled by the Markov Chain $\{Q_{\overline{n}}^{\overline{k}}(D_y)\}$, $\overline{k} = 1, 2, \ldots, M+1$, $\overline{n} = 1, 2, \ldots, \overline{N}(\overline{k})$. Lemma 8 shows that the vector that contains the probabilities that the SGHC algorithm is at solution $\omega_i \in \Omega_y$, i=1, 2, …, $|\Omega_y|$ during the last inner loop iteration N(k+M) of outer loop iteration k+M is

$$f^{(1, \sum_{k=1}^{M+1} \overline{N}(k))} = f^{(0)} P_y(1)^{\overline{N}(1)} P_y(2)^{\overline{N}(2)} \ldots P_y(M+1)^{\overline{N}(M+1)}, \text{ for some } f^{(0)}.$$

Lemma 7

Consider an application of the SGHC algorithm where the problem generation probability function satisfies Criteria A and the SGHC algorithm is defined such that for all outer loop iterations k, if $\Psi(k)\neq\Psi(k-1)$, then the hill climbing random variable, the generation probability function and the inner loop bounds are reset (i.e., $R_k=\overline{R}_1$, $g_{ij}(k)=g_{ij}(1)$, $N(k)=\overline{N}(1)$).

If $y=\Psi(k)\neq\Psi(k-1)$ and $\Psi(k')=y$, for $k'=k, k+1, \ldots, k+M$ and for some $M\in Z^+$, then the SGHC algorithm can be modeled by the Markov Chain $\{Q_{\overline{n}}^{\overline{k}}(D_y)\}$, $\overline{k}=1, 2, \ldots, M+1$, $\overline{n}=1, 2, \ldots, \overline{N}(\overline{k})$, during outer loop iterations k, k+1, …, k+M.

Proof:

Consider the Markov chain $\{Q_{\overline{n}}^{\overline{k}}(D_y)\}$, $\overline{k}=1, 2, \ldots,$ and $\overline{n}=1, 2, \ldots, \overline{N}(\overline{k})$ corresponding to the underlying GHC algorithm executing over the solution space of D_y, y=1, 2, …, m. Recall, from Sections 5.1 and 5.2 that $\overline{k}$ corresponds to the GHC algorithm's hill climbing random variable $\overline{R}_{\overline{k}}$ and the generation probability function $g_{ij}(k)$. Since $\Psi(k')=y$, for $k'=k, k+1, \ldots, k+M$, for some y=1, 2, …, m, then during outer loop iterations k, k+1, …, k+M the SGHC algorithm can be modeled by the Markov Chain $Q_{\overline{n}}^{\overline{k}}(D_y)\}$, where (since the hill climbing random variable, the generation probability function and the inner loop bounds are reset (i.e., $R_k=\overline{R}_1$, $g_{ij}(k)=g_{ij}(1)$, $N(k)=\overline{N}(1)$), $\overline{k}=1, 2, \ldots, M+1$ and $\overline{n}=1, 2, \ldots, \overline{N}(\overline{k})$.

□

Lemma 8

Consider an application of the SGHC algorithm where the problem generation probability function satisfies Criteria A and the SGHC algorithm is defined such that for all outer loop iterations k, if $\Psi(k)\neq\Psi(k-1)$, then the hill climbing random variable,

the generation probability function and the inner loop bounds are reset (i.e., $R_k = \overline{R}_1$, $g_{ij}(k) = g_{ij}(1)$, $N(k) = \overline{N}(1)$).

If $y = \Psi(k) \neq \Psi(k-1)$ and $\Psi(k') = y$, for $k' = k, k+1, \ldots, k+M$, for some $M \in Z^+$, then the vector that defines the probability that the SGHC is in solution $\omega_i \in \Omega_y$, $i = 1, 2, \ldots, |\Omega_y|$ at the last inner loop iteration $N(k+M)$ of outer loop iteration $k+M$ is

$$f^{(1, \sum_{k=1}^{M+1} \overline{N}(k))} = f^{(0)} P_y(1)^{\overline{N}(1)} P_y(2)^{\overline{N}(2)} \ldots P_y(M+1)^{\overline{N}(M+1)}, \text{ for some } f^{(0)}.$$

Proof:

From Lemma 7, during outer loop iterations k, k+1, …, k+M, the SGHC algorithm can be modeled by the Markov Chain $\{Q_{\overline{n}}^{\overline{k}}(D_y)\}$, $\overline{k} = 1, 2, \ldots, M+1$, $\overline{n} = 1, 2, \ldots, \overline{N}(\overline{k})$. Therefore, the corresponding transition matrices are

$$P_y(1) \text{ for } \overline{k} = 1, \overline{n} = 1, 2, \ldots, \overline{N}(1),$$
$$P_y(2) \text{ for } \overline{k} = 2, \overline{n} = 1, 2, \ldots, \overline{N}(2),$$
$$\vdots$$
$$P_y(M+1) \text{ for } \overline{k} = M+1, \overline{n} = 1, 2, \ldots, \overline{N}(M+1).$$

Therefore, the vector that defines the probability that the SGHC algorithm is in solution $\omega_i \in \Omega_y$, $i = 1, 2, \ldots, |\Omega_y|$ at the last inner loop iteration $N(k+M)$ of outer loop iteration $k+M$ is

$$f^{(1, \sum_{k=1}^{M+1} \overline{N}(k))} = f^{(0)} P_y(1)^{\overline{N}(1)} P_y(2)^{\overline{N}(2)} \ldots P_y(M+1)^{\overline{N}(M+1)}, \text{ for some } f^{(0)}.$$

□

Theorem 12 presents sufficient conditions that guarantee that a SGHC will visit every globally optimal solution in every discrete optimization problem in S.

Theorem 12

Consider an application of the SGHC algorithm where the problem generation probability function satisfies Criteria A and the SGHC algorithm is defined such that

for all outer loop iterations k, if $\Psi(k) \neq \Psi(k-1)$, then the hill climbing random variable, the generation probability function and the inner loop bounds are reset (i.e., $R_k = \overline{R}_1$, $g_{ij}(k) = g_{ij}(1)$, $N(k) = \overline{N}(1)$). If the underlying GHC algorithm converges to the set of globally optimal solutions, $G_y \subseteq \Omega_y$, for every $D_y = (\Omega_y, f_y) \in S$, then for all $\varepsilon > 0$, there exists a finite iteration such that the SGHC algorithm visits G_y with probability greater than $1-\varepsilon$.

Proof:

Since the GHC algorithm converges for every $D_y = (\Omega_y, f_y) \in S$, then the Markov chain $\{Q_{\overline{n}}^{\overline{k}}(D_y)\}$ is strongly ergodic for every D_y, y=1, 2, …, m (Isaacson and Madsen 1985). Therefore,

$$\lim_{n \to +\infty} \sup_{f^{(0)}} \| f^{(1, \sum_{k=1}^{M+1} \overline{N}(k))} - \mathbf{q} \| = 0,$$

where $q_i = 0$, for all i =1, 2, …, $|\Omega_y|$ that do not correspond to the globally optimal solution and $\sum_{g \in G_y} q_g = 1$.

Therefore, for all $\varepsilon > 0$, there exists a $M(\varepsilon) \in Z^+$ such that for all outer loop iterations $M \geq M(\varepsilon)$,

$$\sup_{f^{(0)}} \| f^{(1, \sum_{k=1}^{M+1} \overline{N}(k))} - \mathbf{q} \| < \varepsilon.$$

From Theorem 11, there exists an outer loop iteration j such that the SGHC algorithm is in discrete optimization problem D_y at outer loop iterations j, j+1, …, j+M(ε). Let

$$k = \max \{k' : k' \leq j \text{ and } \Psi(k'-1) \neq y\},$$

then for outer loop iterations k, k+1, …, j, j+1, …, j+M(ε), the SGHC algorithm can be modeled by the Markov chain $\{Q_{\overline{n}}^{\overline{k}}(D_y)\}$, $\overline{k}$ =1, 2, …, M+1, $\overline{n}$ =1, 2, …, $\overline{N}(\overline{k})$, where $M \geq M(\varepsilon)$. Since $M \geq M(\varepsilon)$, then

$$\sup_{f^{(0)}} \| f^{(1, \sum_{k=1}^{M+1} \overline{N}(k))} - \mathbf{q} \| < \varepsilon.$$

Hence, for inner loop iteration $N(j+M(\varepsilon))$ during outer loop iteration $j+M(\varepsilon)$, the SGHC algorithm visits an element of G_y with probability greater than $1-\varepsilon$.

□

Corollary 1 presents sufficient conditions that guarantee that a SGHC algorithm that satisfies Criteria A will visit every globally optimal solution over the set of discrete optimization problem in S.

Corollary 1

Consider an application of the SGHC algorithm where the problem generation probability function satisfies Criteria A and the SGHC algorithm is defined such that for every outer loop iteration k, if $\Psi(k)\neq\Psi(k-1)$, then the hill climbing random variable, the generation probability function and the inner loop bounds are reset (i.e., $R_k=\overline{R}_1$, $g_{ij}(k)=g_{ij}(1)$, $N(k)=\overline{N}(1)$). Let G_y be the set of globally optimal solutions over the set of discrete optimization problems in S (i.e., for all $\omega\in G_y$, $f_y(\omega)=\min\{f_i(\omega_i)$: for all i=1, 2,…, m, for all $\omega_i\in G_i\}$. If the underlying GHC algorithm converges to the set of globally optimal solution G_j for every $D_j=(\Omega_j, f_j)\in S$, then for all $\varepsilon>0$, there exists an iteration such that the SGHC algorithm visits an element of G_y with probability greater than $1-\varepsilon$.

Proof:

Immediately follows from Theorem 12.

□

Chapter 8:

Illustrative Example

This chapter contains an illustrative example of a set of fundamentally related discrete optimization problems. The chapter is organized as follows. Section 8.1 presents the Traveling Salesman Problem and formulates it as a discrete optimization problem. Section 8.2 presents the Multiple Traveling Salesman Problem and formulates it as a set of fundamentally related discrete optimization problems

8.1 The Traveling Salesman Problem

The traveling salesman problem (TSP) is a well-known NP-hard discrete optimization problem (Lawler 1985). The TSP is used to illustrate various local search algorithms because it is useful for modeling a variety of real world problems. For instance, traditional applications of the TSP include a variety of vehicle routing and scheduling problems. More recently, applications of the TSP have been expanded to include modern applications like the printing of circuit boards, x-ray crystallography, overhauling of gas turbine engines, and the controlling of industrial robots (Johnson and Jacobson 2000b).

To formally define the TSP the following definitions are needed (Lawler 1985). Define a *graph* to be a finite set of vertices, some pairs of which are joined by edges. A *cycle* in a

graph is a set of vertices of the graph, which is such that it is possible to move from vertex to vertex, along edges of the graph, so that all vertices are encountered exactly once, finishing at the start. If a cycle contains all the vertices of the graph, it is called a *Hamiltonian cycle (or tour)*. The TSP is defined as follows (Garey and Johnson 1979).

Instance: Given a set of n cities $C=\{c_1, c_2, \ldots, c_n\}$ and a distance matrix D that represents the cost of traveling between the cities in the set C.

Question: Find a Hamiltonian cycle $H=(c_1, c_2, \ldots, c_n)$ such that

$$f(H)=\sum_{j=1}^{n-1} D(c_j, c_{j+1}) + D(c_n, c_1)$$

is minimized.

An instance of a TSP is a discrete optimization problem, where the solution space is the set of possible all Hamiltonian cycles (with each tour consisting of n cities), $\Omega=\{\omega_1, \omega_2, \ldots, \omega_{\frac{(n-1)!}{2}}\}$. The objective function value for each solution $\omega_i=(c_1, c_2, \ldots, c_n)\in\Omega$ is the sum of the distances the tour depicts, $f(\omega_i)=\sum_{j=1}^{n-1} D(c_j, c_{j+1}) + D(c_n, c_1)$. The optimal objective function value represents the least distance traveled. A common neighborhood function used for the TSP is the 2-Opt neighborhood function.

The 2-Opt neighborhood function for the TSP can be described as moving from one solution ω_p to another solution ω_q by the exchange of two edges. For example, consider the finite set of seven cities $C=\{c_1, c_2, c_3, c_4, c_5, c_6, c_7\}$, and the corresponding solution space $\Omega=\{\omega_1, \omega_2, \ldots, \omega_{60}\}$. Figure 8.1 illustrates the 2-Opt neighborhood moving from solution $\omega_1=(c_1, c_2, c_3, c_4, c_5, c_6, c_7)$ to solution $\omega_2=(c_1, c_5, c_4, c_3, c_2, c_6, c_7)$.

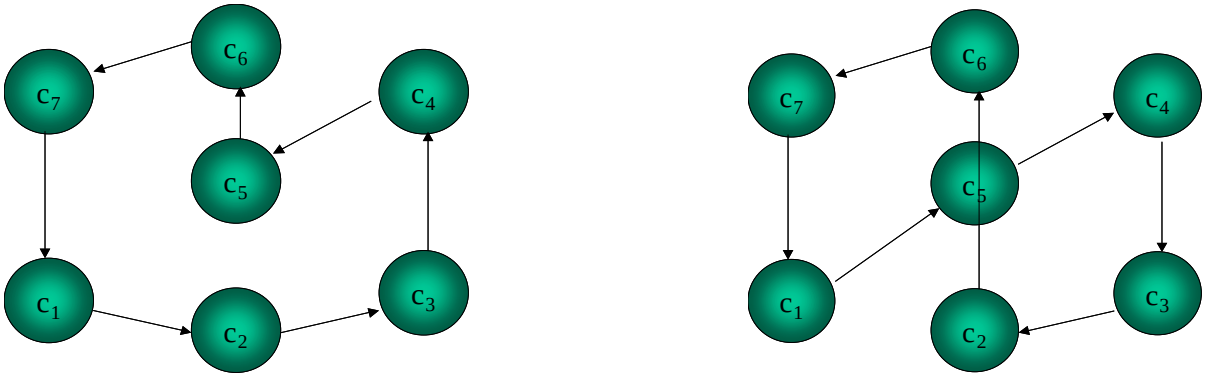

Figure 8.1: The 2-Opt Neighborhood Function

For a complete discussion of the 2-Opt neighborhood function, see Aarts and Lenstra (1997).

The city exchange neighborhood function for the TSP can be described as moving from one solution ω_p to another solution ω_q by the exchange of two cities. For example, consider the finite set of seven cities $C=\{c_1, c_2, c_3, c_4, c_5, c_6, c_7\}$, and the corresponding solution space $\Omega=\{\omega_1, \omega_2, \ldots, \omega_{60}\}$. Figure 8.2 illustrates the city exchange neighborhood function moving from solution $\omega_1=(c_1, c_2, c_3, c_4, c_5, c_6, c_7)$ to solution $\omega_2= (c_1, c_5, c_3, c_4, c_2, c_6, c_7)$.

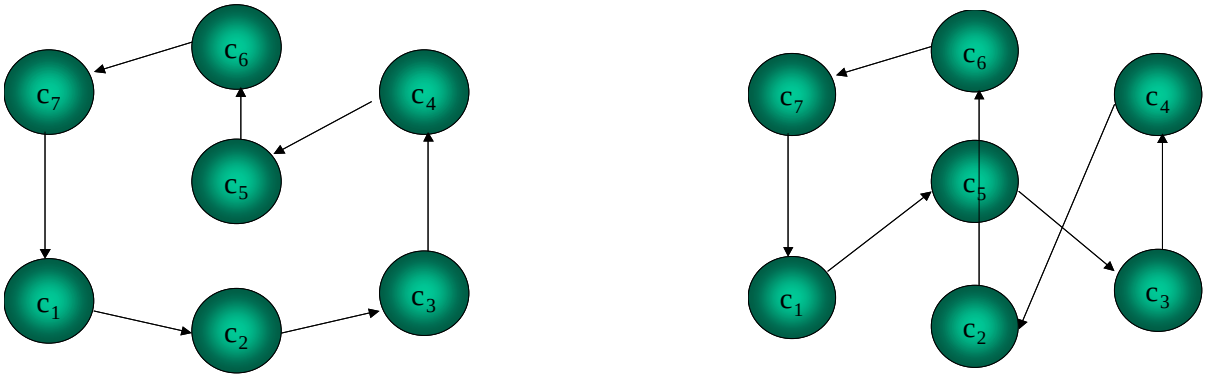

Figure 8.2: The City Exchange Neighborhood Function

8.2 The Multiple Traveling Salesman Problem

This section considers simultaneously approaching several traveling salesman problems using SGHC algorithms. Specifically, the Multiple Traveling Salesman Problem (MTSP) is considered. The MTSP is defined as follows.

Instance: Given a set of n cities $Ob=\{c_1, c_2, \ldots, c_n\}$, a set of m subsets of *Ob*, $O=\{C_1, C_2, ..., C_m\}$, and a distance matrix *D* that represents the cost of traveling between the cities in the set *Ob*.

Question: Find a Hamiltonian cycle $H=(c_1, c_2, \ldots, c_{|\Omega_y|})$ where there exists a C_y, y=1, 2, …, m such that $c_j \in C_y$, for every j=1, 2, …, n, and

$$f(H)=\sum_{j=1}^{n-1} D(c_j, c_{j+1}) + D(c_n, c_1)$$

is minimized.

Note that each of the sets $C_y \in O$ represents an instance of the TSP, D_y. Section 8.1 formulated the TSP as a discrete optimization problem. Therefore, the MTSP problem can be represented by set of discrete optimization problems $S=\{D_1, D_2, ..., D_m\}$. The set $S=\{D_1, D_2, ..., D_m\}$ is a set of fundamentally related discrete optimization problems. To see this, note that each discrete optimization problem $D_y \in S$, y=1, 2, …, m is fully defined by $C_y \subseteq Ob=\{c_1, c_2, \ldots, c_n\}$. Therefore, C_y is the fundamental relation set of discrete optimization problem D_y.

8.2.1 Illustrative Example of the Multiple Traveling Salesman Problem

Consider the instance of the MTSP problem defined in the following tables. Table 9-1 contains the set *Ob* of cities and Table 8-2 contains the set of fundamental relation

sets, O={C_1, C_2, ..., C_m}, $C_y \subseteq Ob$. Then the TSP's, D_y, y=1, 2, 3, 4, can be represented by the binary activity vectors $\mathbf{c}^y \in \{0, 1\}^n$ in Table 8-3.

Table 8-1: The Set of Objects, *Ob*

Ob={Minneapolis, Washington D.C., Chicago, Los Angeles, Seattle, San Francisco, New York, Pittsburgh, Detroit, Dallas, Boulder}

Table 8-2: Fundamental Relation Set

TSP Instance	Fundamental Relation Set
D_1	C_1={Washington D.C., Chicago, Los Angeles, Seattle, San Francisco, New York, Boulder}
D_2	C_2={Minneapolis, Washington D.C., Chicago, Los Angeles, Seattle, New York, Pittsburgh, Dallas, Boulder}
D_3	C_4={Minneapolis, Washington D.C., Chicago, Los Angeles, Seattle, New York, Detroit, Boulder}
D_4	C_4={Minneapolis, Washington D.C., Chicago, Los Angeles, Seattle, San Francisco, New York, Pittsburgh, Boulder}

Table 8-3: Binary Activity Vectors

Option	$\mathbf{c}^y = (B_1, B_2, \ldots, B_{11})$
$\mathbf{c}^1$	(0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1)
$\mathbf{c}^2$	(1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1)
$\mathbf{c}^3$	(1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1)
$\mathbf{c}^4$	(1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1)

The detachment metric $\rho(D_y, D_q)$, for every y,q=1, 2, 3, 4 can be depicted by the distance matrix and distance diagram in Figure 8.3.

	D_1	D_2	D_3	D_4
D_1	0	4	3	2
D_2	4	0	3	2
D_3	3	3	0	3
D_4	2	2	3	0

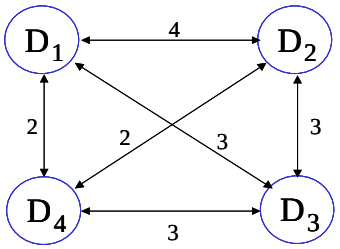

Figure 8.3: Distance Matrix and Distance Diagram

Chapter 9:

Computational Results

This chapter provides computational results for addressing a MTSP with SGHC algorithms. For comparison purposes, GHC algorithms are also applied to the individual members in the set of fundamentally related discrete optimization problems. These computational results suggest that optimal/near optimal solutions can be reached in less total iterations using a SGHC algorithm.

To develop a MTSP, a set consisting of 20 cities was generated by randomly locating each city on a 1000 by 1000 unit grid. Four TSPs of size 18 were generated by randomly selecting 18 of the 20 cities for each traveling salesman problem. In the case where an identical set of cities was generated for two or more of the problems, a completely new set of discrete optimization problems was generated, ensuring four distinct randomly generated TSPs.

The four randomly generated TSPs are arbitrarily labeled D_1, D_2, D_3, and D_4. The detachment metric $\rho(D_y, D_q)$, between the TSPs, was calculated for all y,q=1, 2, 3, 4. The distance matrix and distance diagram are depicted in Figure 9.1.

	D_1	D_2	D_3	D_4
D_1	0	4	4	4
D_2	4	0	4	4
D_3	4	4	0	2
D_4	4	4	2	0

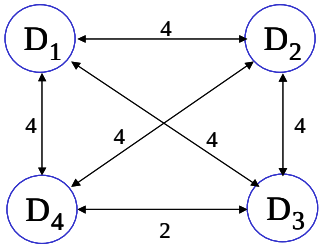

Figure 9.1: Distance Matrix and Distance Diagram

9.1 Stationary Markov Chain Computational Results

Computational results with Monte Carlo search, pure local search, and simulated annealing using SGHC algorithms are reported. For comparison purposes, computational results with Monte Carlo search, pure local search, and simulated annealing using GHC algorithms are also reported. The 2-Opt neighborhood function was used for all executions of the SGHC and GHC algorithms. For the SGHC algorithms, the problem generation probability function was defined as

$$h_{D_y D_q}(k, \rho(D_y, D_q)) = [1/\rho(D_y, D_q)] / [\sum_{\substack{i=1 \\ i \neq y}}^{4} (1/\rho(D_y, D_i))], \; y \neq q$$

and

$$h_{D_y D_y}(k, \rho(D_y, D_y)) = 1 - \sum_{\substack{q=1 \\ q \neq y}}^{4} [1/(\rho(D_y, D_q)] / [\sum_{\substack{i=1 \\ i \neq y}}^{4} (1/\rho(D_y, D_i))] = 0,$$

for every y,q= 1, 2, 3, 4, y≠q for every k=1, 2, …, K.

This problem generation probability function is such that the associated transition matrix is symmetric and the Markov chain {Ψ(k)} is stationary. Therefore, by Theorem 1, the Markov chain {Ψ(k)} has a uniform stationary distribution, hence as k approaches infinity, the SGHC algorithm is executing over the solution space of each discrete optimization problem in S

$=\{D_1, D_2, \ldots, D_m\}$ with probability $1/m=1/|S|=1/4$. Moreover, this problem generation function guarantees the discrete optimization problem over which the SGHC algorithm is executing changes at every outer loop iteration k (i.e., $\Psi(k)\neq\Psi(k-1)$, for all k=1, 2, …).

Executions with different values of K and N=N(k), k = 1, 2, …, K are reported. To compare the performance of applying a SGHC algorithm versus applying a GHC algorithm, the inner and outer loop bounds of the SGHC algorithm were doubled. Therefore, the total number of iterations that the SGHC algorithm executes is equal to the total number of iterations executed using the GHC algorithm for the four individual problems. Let $R\in Z^+$ represent the total number of replications executed for each SGHC and GHC algorithm formulation. For each replication, a different randomly generated initial tour was used. The means, μ, standard deviations, σ, and the minimum and maximum distances, were computed from the optimal tour distances across these R replications. The value γ in Tables 9-1 through 9-6 represents the number of replications for which the algorithms find the minimum distance tour. For simulated annealing, t_k is updated by multiplying the previous temperature parameter by the increment multiplier β=.90 (i.e., $t_k=\beta t_{k-1}$). The initial temperature parameter is t_0.

Table 9-1: GHC Algorithm Results: Pure Local Search

Inner and Outer Loop Bounds	γ/R	μ	σ	Minimum	Maximum
K=100, M=100	4/30	3864.3	36.9726	3805.4	3916.0
K=200, M=100	3/30	3863.4	32.7691	3805.4	3907.3
K=300, M=75	1/30	3876.6	36.9549	3805.4	3953.7
K=400, M=75	1/30	3885.3	42.1893	3805.4	3973.4
K=400, M=50	2/30	3867.9	37.7520	3805.4	3916.0
K=800, M=50	1/15	3872.8	35.2469	3805.4	3916.0

Table 9-2: SGHC Algorithm Results: Pure Local Search

Inner and Outer Loop Bounds	γ/R	μ	σ	Minimum	Maximum
K=100, M=100	10/30	3819.4	13.2943	3805.4	3831.8
K=200, M=100	23/30	3808.9	9.0535	3805.4	3831.6
K=300, M=75	23/30	3808.9	9.0535	3805.4	3831.6
K=400, M=75	24/30	3807.2	6.6434	3805.4	3831.6
K=400, M=50	9/30	3818.5	13.3165	3805.4	3831.6
K=800, M=50	12/15	3810.7	10.8417	3805.4	3831.6

Table 9-1 suggests that when the number of outer loop iterations for a pure local search GHC algorithm is increased from 100 to 200, performance of the algorithm shows no improvement. However, Table 9-2 suggests that the performance of a pure local search SGHC algorithm improves significantly when the number of outer loop iterations is increased from 100 to 200, as measured by μ.

Table 9-3: GHC Algorithm Results: Simulated Annealing

Inner and Outer Loop Bounds	t_0	γ/R	μ	σ	Minimum	Maximum
K=100, M=100	3000	1/30	3854.1	41.4192	3805.4	3953.7
K=200, M=100	3000	1/30	3861.2	35.7662	3805.4	3916.0
K=300, M=75	2000	2/30	3861.5	39.7074	3805.4	3973.4
K=400, M=75	2000	5/30	3855.9	35.3872	3805.4	3907.5
K=400, M=50	2000	3/30	3868.8	39.4863	3805.4	3916.0
K=800, M=50	2000	1/15	3885.9	29.7020	3805.4	3916.0

Table 9-4: SGHC Algorithm Results: Simulated Annealing

Inner and Outer Loop Bounds	t_0	γ/R	μ	σ	Minimum	Maximum
K=100, M=100	3000	16/30	3814.1	12.5549	3805.4	3831.6
K=200, M=100	3000	19/30	3808.0	7.9899	3805.4	3831.6
K=300, M=75	2000	20/30	3808.9	9.0535	3805.4	3831.6
K=400, M=75	2000	23/30	3808.9	9.0535	3805.4	3831.6
K=400, M=50	2000	7/30	3831.6	13.1248	3805.4	3831.6
K=800, M=50	2000	1/15	3813.6	12.5352	3805.4	3831.6

Table 9-5: GHC Algorithm Results: Monte Carlo Search

Inner and Outer Loop Bounds	γ/R	μ	σ	Minimum	Maximum
K=100, M=100	1/30	6390.0	294.6998	5634.9	6805.9
K=200, M=100	1/30	6195.6	239.7325	5575.7	6656.3
K=300, M=75	1/30	6111.7	293.6761	5293.0	6613.9
K=400, M=75	1/30	6099.6	292.5790	5310.2	6488.3
K=400, M=50	1/30	6122.6	287.3693	5291.7	6575.7
K=800, M=50	1/15	6007.1	231.746	5479.1	6328.1

Table 9-6: SGHC Algorithm Results: Monte Carlo Search

Inner and Outer Loop Bounds	γ/R	μ	σ	Minimum	Maximum
K=100, M=100	1/30	6758.2	202.9184	5793.3	6758.2
K=200, M=100	1/30	6109.3	301.5243	5243.7	6611.5
K=300, M=75	1/30	6214.6	204.3092	5727.6	6575.3
K=400, M=75	1/30	6054.3	232.6718	5614.9	6462.8
K=400, M=50	1/30	6265.8	197.6956	5877.3	6659.3
K=800, M=50	1/15	5954.7	301.7151	5299.6	6382.3

Tables 9-5 and 9-6 suggest that there is little difference in performance between Monte Carlo search GHC algorithms and Monte Carlo search SGHC algorithms. Overall, Tables 9-1 through 9-4 suggest that the SGHC algorithms outperform the GHC algorithms. The minimum distance found over the R replications using SGHC algorithms is significantly smaller than the minimum distance found over the R replications using GHC algorithms for both the simulated annealing and pure local search algorithms. Additionally, the standard deviation of the optimal values over the R replications is much smaller using the SGHC algorithms.

Figures 9.2 through 9.4 depict plots comparing the performance of the SGHC algorithms and the GHC algorithms. To obtain this data, fifteen replications of each SGHC algorithm and GHC algorithm formulation were executed. For each replication, a different randomly generated initial solution was used. The mean of the optimal distances across the fifteen replications for the GHC algorithm are plotted with a solid blue line. The standard deviations of the optimal distances for the GHC algorithm across the fifteen replications are plotted with a dashed blue line. The means of the optimal distances across the fifteen replications for the SGHC algorithm are plotted with a solid red line. The standard deviations of the optimal distances for the SGHC algorithm across the fifteen replications are plotted with a dashed red line. The number of outer loop iterations executed was 800 and the number of inner loop iterations executed was 50 for every formulation.

For simulated annealing, t_k is updated by multiplying the previous temperature parameter by the increment multiplier β=.90 with initial temperature parameter t_0=2,000.

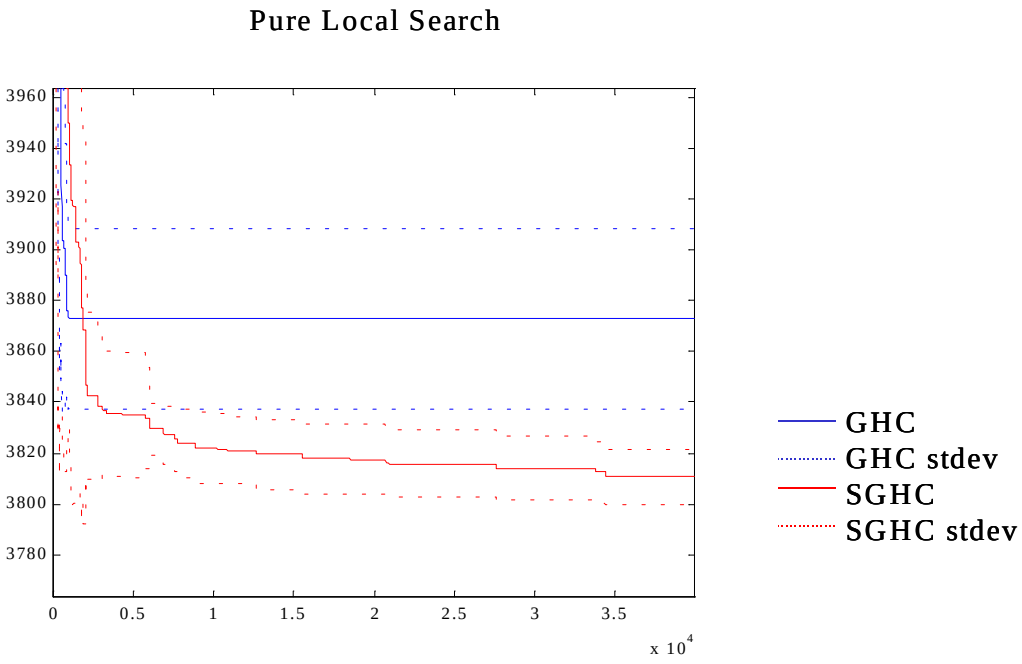

Figure 9.2: Pure Local Search

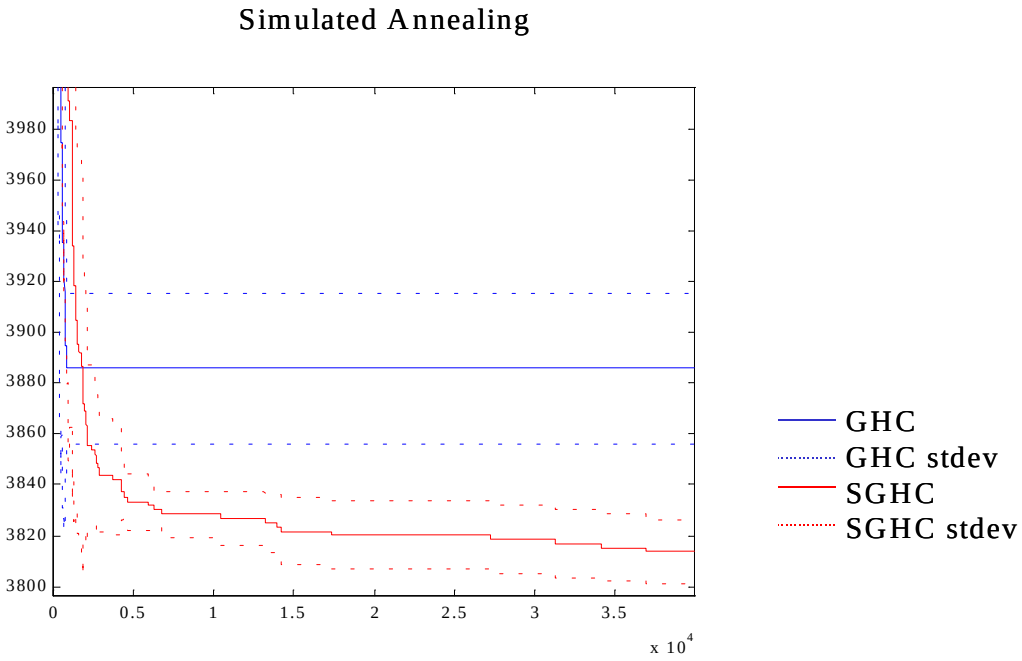

Figure 9.3: Simulated Annealing

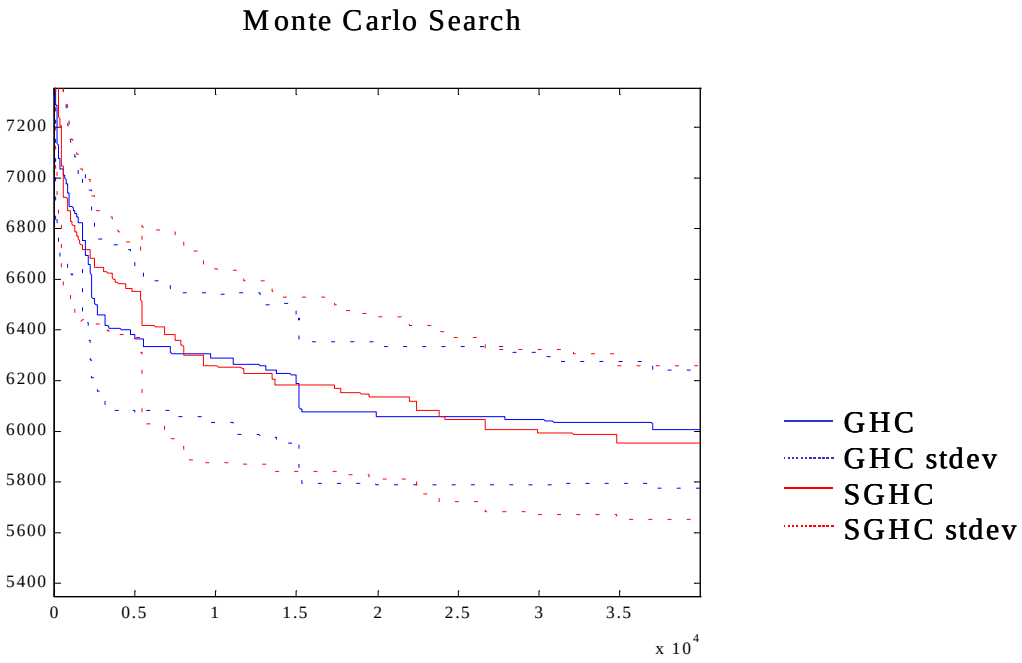

Figure 9.4: Monte Carlo Search

Figures 9.2 and 9.3 suggest that the SGHC algorithms outperform the GHC algorithms, as measured by μ. The minimum distance found over the fifteen replications using SGHC algorithms is significantly smaller after 2500 iterations than the minimum distance found over the fifteen replications using GHC algorithms for both the simulated annealing and pure local search algorithms. Moreover, the standard deviation band of the optimal values over the fifteen replications is much smaller using SGHC algorithms. Figure 9.4 suggests that there is no significant difference in the performance of Monte Carlo Search SGHC and GHC algorithms.

Figures 9.5 through 9.7 depict plots comparing the performance of the SGHC algorithms and the GHC algorithms. To obtain this data, thirty replications of each SGHC algorithm and GHC algorithm formulation were executed. For each replication, a different randomly

generated initial solution was used. The mean of the optimal distances across the thirty replications for the GHC algorithm are plotted with a solid blue line. The standard deviations of the optimal distances for the GHC algorithm across the thirty replications are plotted with a dashed blue line. The means of the optimal distances across the thirty replications for the SGHC algorithm are plotted with a solid red line. The standard deviations of the optimal distances for the SGHC algorithm across the thirty replications are plotted with a dashed red line. The number of outer loop iterations executed was 400 and the number of inner loop iterations executed was 75 for every formulation.

For simulated annealing, t_k is updated by multiplying the previous temperature parameter by the increment multiplier β=.90 with initial temperature parameter t_0=2,000.

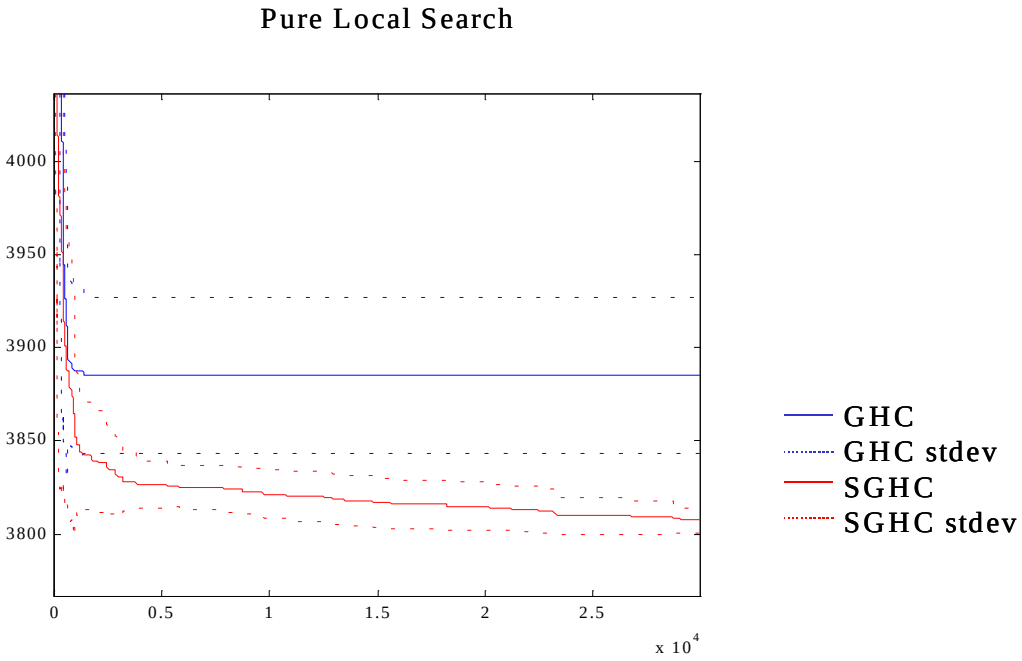

Figure 9.5: Pure Local Search

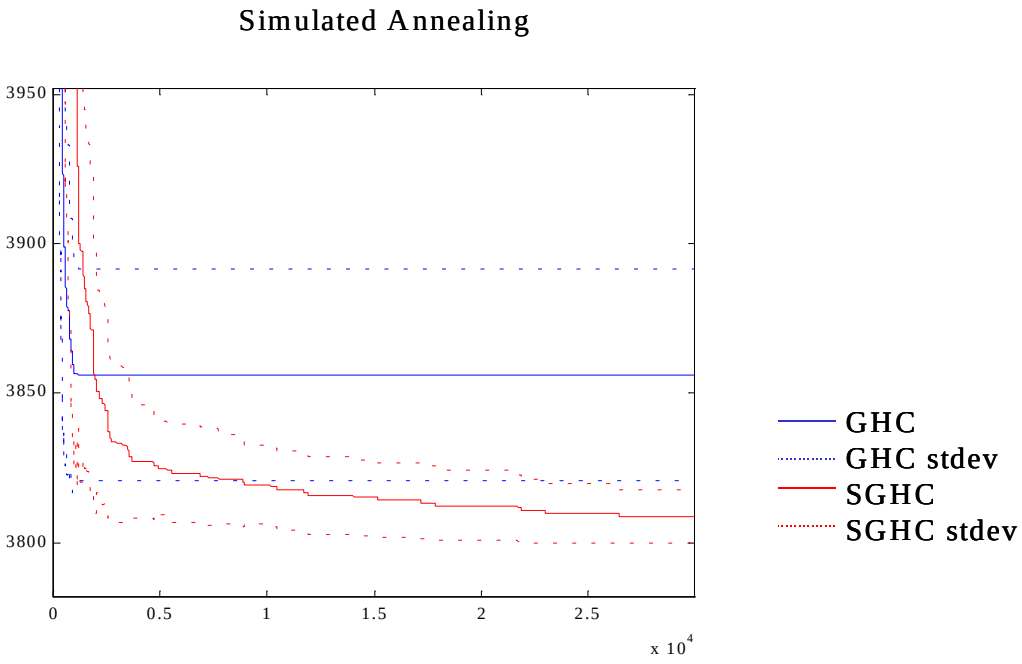

Figure 9.6: Simulated Annealing

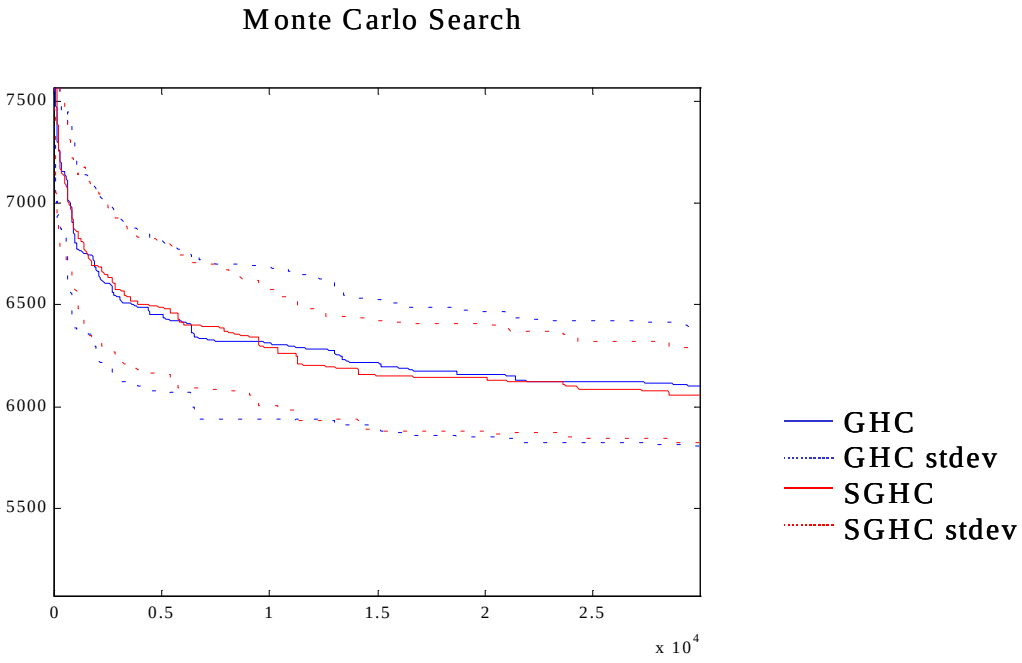

Figure 9.7: Monte Carlo Search

Figures 9.5 and 9.6 suggest that the SGHC algorithms outperform the GHC algorithms, as measured by μ. The minimum distance found over the thirty replications using SGHC algorithms is significantly smaller after 2500 iterations than the minimum distance found over the thirty replications using GHC algorithms for both the simulated annealing and pure local search algorithms. Moreover, the standard deviation band of the optimal values over the thirty replications is much smaller using SGHC algorithms. Figure 9.7 suggests that there is no significant difference in the performance of Monte Carlo Search SGHC and GHC algorithms.

9.2 Nonstationary Markov Chain Computational Results

Computational results with Monte Carlo search, pure local search, and simulated annealing using SGHC algorithms are reported. For comparison purposes, computational results with Monte Carlo search, pure local search, and simulated annealing using GHC algorithms are also reported. The 2-Opt neighborhood function was used for all executions of the SGHC and GHC algorithms. For the SGHC algorithms, the problem generation probability function was defined as follows.

$h_{D_y D_y}(k, \rho(D_y, D_y)) = 1\text{-}1/(k+1)$ for every k=1, 2, …, K and for every y=1, 2, …, m

and

$$h_{D_y D_q}(k, \rho(D_y, D_q)) = [1/(\rho(D_y, D_q)(k+1))] / [\sum_{\substack{i=1 \\ i \neq y}}^{4} (1/\rho(D_y, D_i))],$$

for every y,q= 1, 2, 3, 4 y≠q.

This problem generation probability function ensures that the Markov chain {Ψ(k)} is weakly ergodic (see Theorem 7). Additionally, this problem generation probability function satisfies Criteria A. Therefore, for this SGHC algorithm, the results in Section 7.1 and Section 7.2 hold.

The SGHC algorithm is defined such that for every outer loop iteration k, if $\Psi(k) \neq \Psi(k-1)$, then the hill climbing random variable, the generation probability function and the inner loop bounds are reset. Therefore, for this SGHC algorithm, if the underlying GHC algorithm converges, then the convergence conditions in Corollary 1 (of Theorem 12) are satisfied.

Executions with different values of K and N=N(k), k = 1, 2, …, K are reported. To compare the performance of applying a SGHC algorithm versus applying a GHC algorithm, the inner and outer loop bounds of the SGHC algorithm were doubled. Therefore, the total number of iterations that the SGHC algorithm executes is equal to the total number of iterations executed using the GHC algorithm for the four individual problems. Let $R \in Z^+$ represent the total number of replications executed for each SGHC and GHC algorithm formulation. For each replication, a different randomly generated initial tour was used. The means, μ, standard deviations, σ, and the minimum and maximum distances, were computed from the optimal tour distances across these R replications. The value γ in Tables 9-7 through 9-11 represents the number of replications for which the algorithms find the minimum distance tour. For simulated annealing, t_k is updated by multiplying the previous temperature parameter by the increment multiplier β=.90 (i.e., $t_k = \beta t_{k-1}$). The initial temperature parameter is t_0.

Table 9-7: GHC Algorithm Results: Pure Local Search

Inner and Outer Loop Bounds	γ/R	μ	σ	Minimum	Maximum
K=100, M=100	4/30	3864.3	36.9726	3805.4	3916.0
K=200, M=100	3/30	3863.7	32.7691	3805.4	3907.3
K=300, M=75	1/30	3876.6	36.9549	3805.4	3953.7
K=400, M=75	1/30	3885.3	42.1893	3805.4	3973.4
K=400, M=50	2/30	3867.9	37.7520	3805.4	3916.0
K=800, M=50	1/15	3872.8	35.2469	3831.8	3916.0

Table 9-8: SGHC Algorithm Results: Pure Local Search

Inner and Outer Loop Bounds	γ/R	μ	σ	Minimum	Maximum
K=100, M=100	9/30	3826.1	18.0558	3805.4	3896.3
K=200, M=100	15/30	3818.6	32.7691	3805.4	3907.3
K=300, M=75	12/30	3821.2	13.1080	3805.4	3832.5
K=400, M=75	16/30	3815.9	13.0973	3805.4	3832.5
K=400, M=50	16/30	3817.7	13.3917	3805.4	3832.5
K=800, M=50	10/15	3814.2	12.8001	3805.4	3831.8

Table 9-7 suggests that when the number of outer loop iterations for a pure local search GHC algorithm is increased from 100 to 200, performance of the algorithm shows no improvement. However, Table 9-8 suggests that the performance of a pure local search SGHC algorithm improves significantly when the number of outer loop iterations is increased from 100 to 200, as measured by μ.

Table 9-9: GHC Algorithm Results: Simulated Annealing

Inner and Outer Loop Bounds	t_0	γ/R	μ	σ	Minimum	Maximum
K=100, M=100	3000	1/30	3878.8	37.5969	3805.4	3953.7
K=200, M=100	3000	5/30	3861.8	38.6953	3805.4	3911.0
K=300, M=75	2000	3/30	3889.0	65.0397	3805.4	4095.9
K=400, M=75	2000	5/30	3860.9	41.0873	3805.4	3865.7
K=400, M=50	2000	3/30	3869.2	59.0710	3805.4	4076.2
K=800, M=50	2000	2/15	3846.4	31.2371	3805.4	3907.3

Table 9-10: SGHC Algorithm Results: Simulated Annealing

Inner and Outer Loop Bounds	t_0	γ/R	μ	σ	Minimum	Maximum
K=100, M=100	3000	7/30	3912.9	116.9697	3805.4	4250.2
K=200, M=100	3000	4/30	3843.6	34.2857	3805.4	3907.3
K=300, M=75	2000	8/30	3833.4	36.2928	3805.4	3973.7
K=400, M=75	2000	15/30	3815.3	15.4484	3805.4	3865.7
K=400, M=50	2000	11/30	3829.5	29.9040	3805.4	3907.3
K=800, M=50	2000	13/15	3807.2	6.7610	3805.4	3816.0

Table 9-11: GHC Algorithm Results: Monte Carlo Search

Inner and Outer Loop Bounds	γ/R	μ	σ	Minimum	Maximum
K=100, M=100	1/30	6390.0	294.6998	5634.9	6805.9
K=200, M=100	1/30	6148.6	286.7861	5389.0	6555.7
K=300, M=75	1/30	6145.0	275.4739	5594.7	6568.4
K=400, M=75	1/30	6099.6	248.263	5479.1	6517.4
K=400, M=50	1/30	6200.6	252.3233	5382.8	6609.0
K=800, M=50	1/15	6041.9	227.0252	5479.1	6517.4

Table 9-12: SGHC Algorithm Results: Monte Carlo Search

Inner and Outer Loop Bounds	γ/R	μ	σ	Minimum	Maximum
K=100, M=100	1/30	6394.6	266.2426	5662.5	6969.3
K=200, M=100	1/30	6157.4	385.8702	5018.2	6600.6
K=300, M=75	1/30	6162.7	207.2497	5635.8	6431.9
K=400, M=75	1/30	6155.1	248.2631	5550.5	6517.4
K=400, M=50	1/30	6163.3	256.5609	5525.1	6599.9
K=800, M=50	1/15	5953.3	205.7178	5434.1	6204.5

Tables 9-11 and 9-12 suggest that there is little difference in performance between Monte Carlo search GHC algorithms and Monte Carlo search SGHC algorithms. Overall, Tables 9-7 through 9-10 suggest that the SGHC algorithms outperform the GHC algorithms. The minimum distance found over the R replications using SGHC algorithms is significantly smaller than the minimum distance found over the R replications using GHC algorithms for both the simulated annealing and pure local search algorithms. Additionally, the standard deviation of the optimal values over the R replications is much smaller using the SGHC algorithms.

Figures 9.8 through 9.10 depict plots comparing the performance of the SGHC algorithms and the GHC algorithms. To obtain this data, fifteen replications of each SGHC algorithm and GHC algorithm formulation were executed. For each replication, a different randomly generated initial solution was used. The mean of the optimal distances across the fifteen replications for the GHC algorithm are plotted with a solid blue line. The standard deviations of the optimal distances for the GHC algorithm across the fifteen replications are plotted with a dashed blue line. The means of the optimal distances across the fifteen replications for the SGHC algorithm are plotted with a solid red line. The standard deviations of the optimal distances for the SGHC algorithm across the fifteen replications are plotted with a dashed red

line. The number of outer loop iterations executed was 800 and the number of inner loop iterations executed was 50 for every formulation.

For simulated annealing, t_k is updated by multiplying the previous temperature parameter by the increment multiplier β=.90 with the initial temperature parameter t_0=2,000.

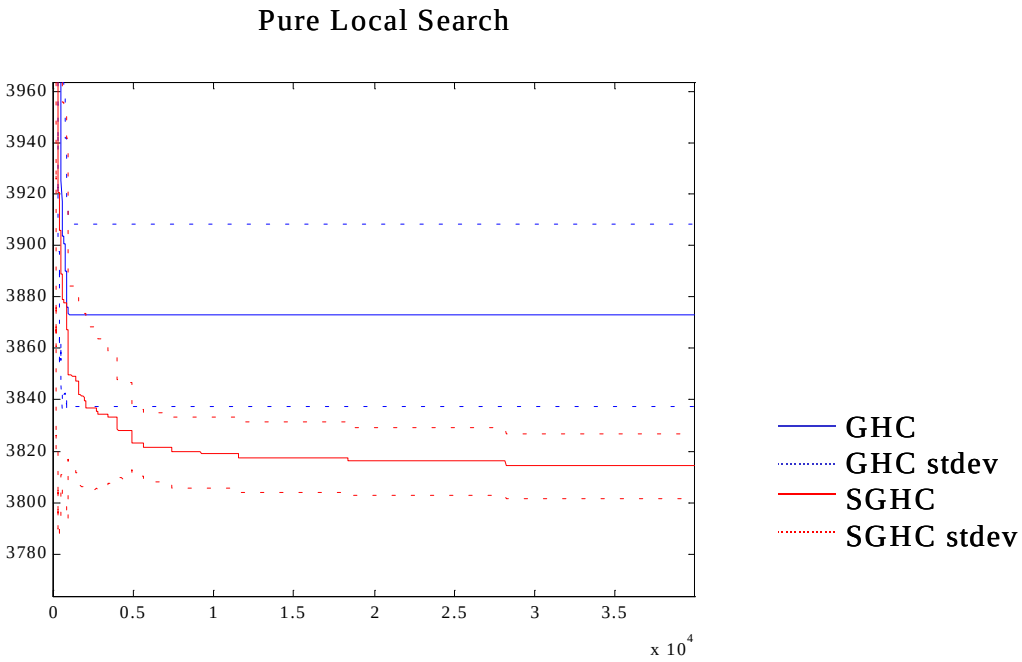

Figure 9.8: Pure Local Search

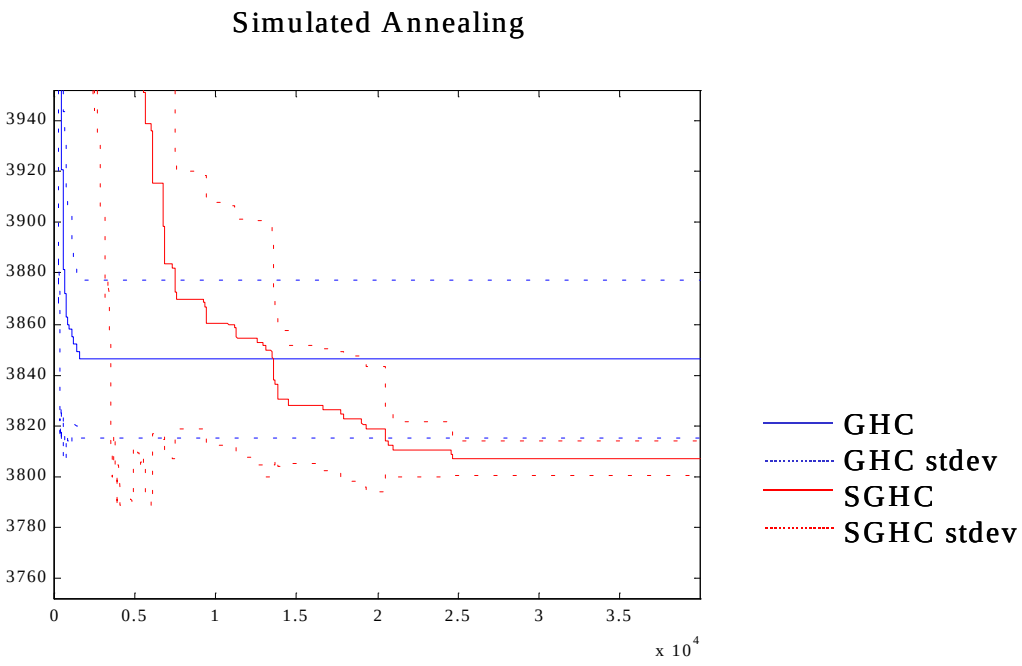

Figure 9.9: Simulated Annealing

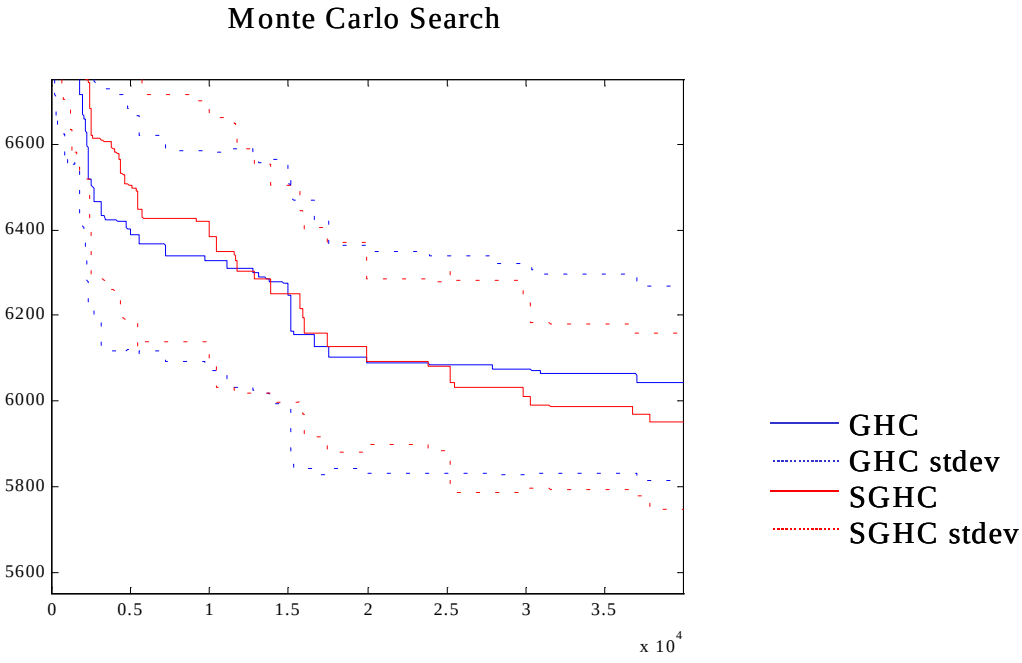

Figure 9.10: Monte Carlo Search

Figures 9.8 and 9.9 suggest that the SGHC algorithms outperform the GHC algorithms, as measured by μ. The minimum distance found over the fifteen replications using SGHC algorithms is significantly smaller after 15000 iterations than the minimum distance found over the fifteen replications using GHC algorithms for both the simulated annealing and pure local search algorithms. Moreover, the standard deviation band of the optimal values over the fifteen replications is much smaller using SGHC algorithms. Figure 9.10 suggests that there is no significant difference in the performance of Monte Carlo Search SGHC and GHC algorithms.

Figures 9.11 through 9.13 depict plots comparing the performance of the SGHC algorithms and the GHC algorithms. To obtain this data, thirty replications of each SGHC algorithm and GHC algorithm formulation were executed. For each replication, a different randomly generated initial solution was used. The mean of the optimal distances across the thirty replications for the GHC algorithm are plotted with a solid blue line. The standard deviations of the optimal distances for the GHC algorithm across the thirty replications are plotted with a dashed blue line. The means of the optimal distances across the thirty replications for the SGHC algorithm are plotted with a solid red line. The standard deviations of the optimal distances for the SGHC algorithm across the thirty replications are plotted with a dashed red line. The number of outer loop iterations executed was 400 and the number of inner loop iterations executed was 75 for every formulation.

For simulated annealing, t_k is updated by multiplying the previous temperature parameter by the increment multiplier β=.90 with initial temperature parameter is t_0=2,000.

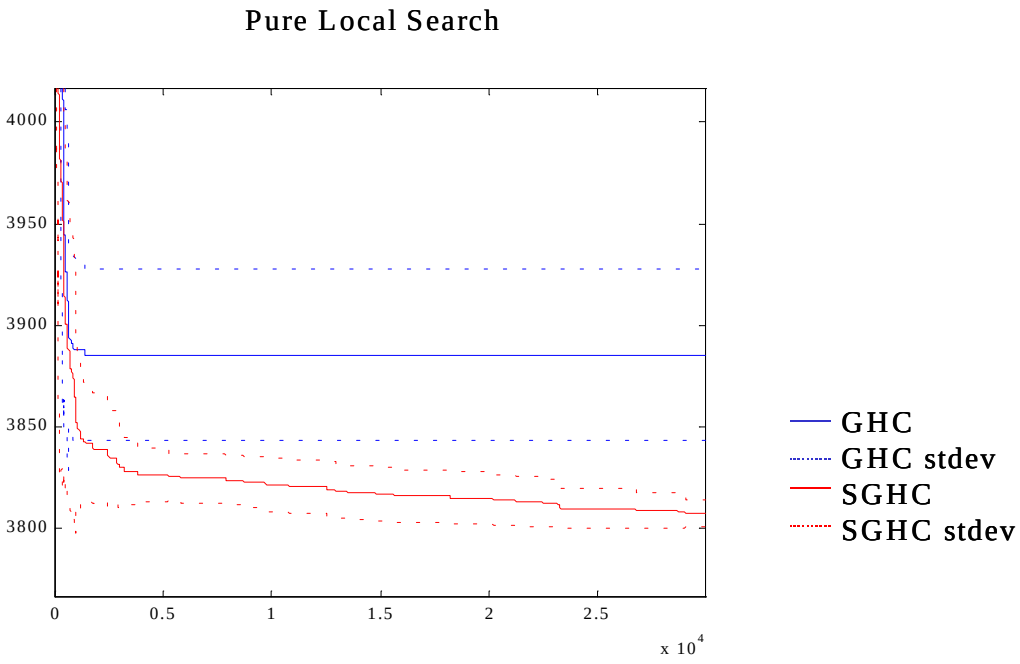

Figure 9.11: Pure Local Search

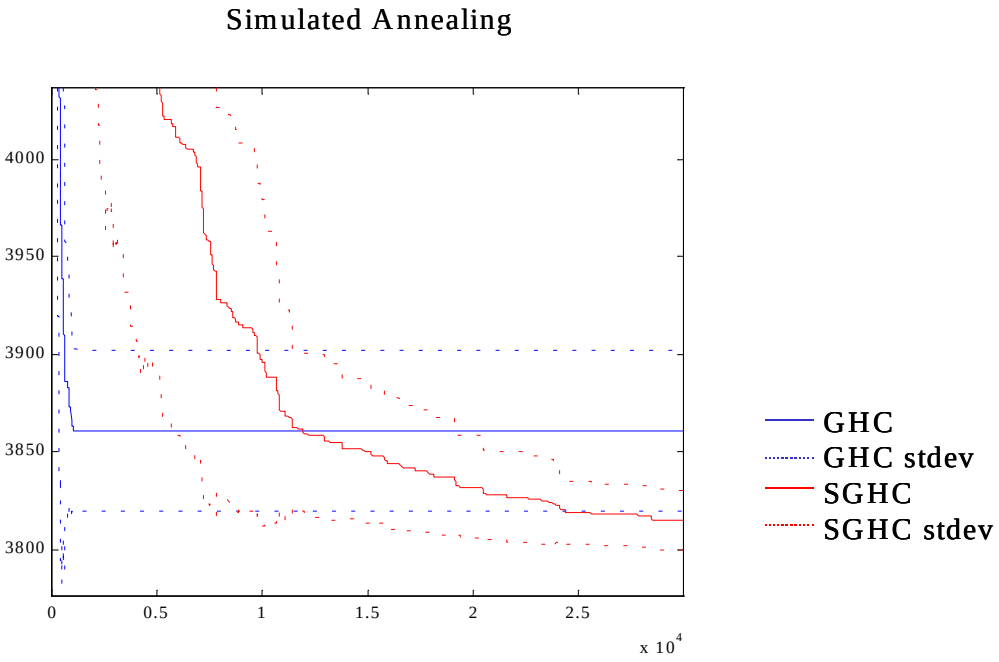

Figure 9.12: Simulated Annealing

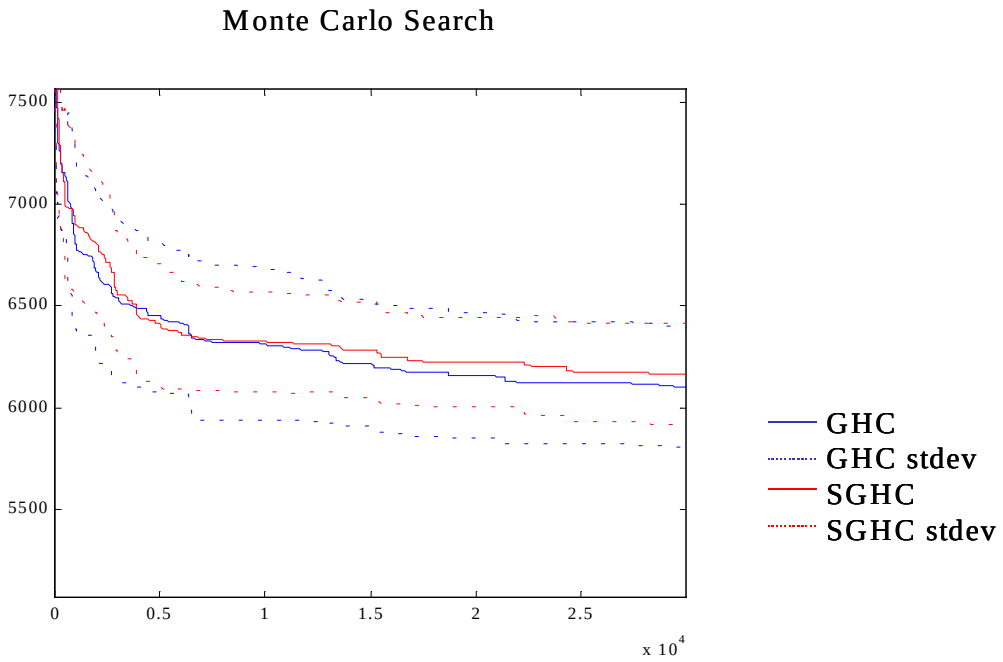

Figure 9.13: Monte Carlo Search

Figures 9.11 and 9.12 suggest that the SGHC algorithms outperform the GHC algorithms, as measured by μ. The minimum distance found over the thirty replications using SGHC algorithms is significantly smaller after 13000 iterations than the minimum distance found over the thirty replications using GHC algorithms for both the simulated annealing and pure local search algorithms. Additionally, the standard deviation band of the optimal values over the thirty replications is much smaller using SGHC algorithms. Figure 9.13 suggests that there is no significant difference in the performance of Monte Carlo Search SGHC and GHC algorithms.

Chapter 10:

Conclusion and Future Directions of Research

A mathematical framework for computationally simultaneously approaching several discrete optimization problems using GHC algorithms is developed and studied in this dissertation. The resulting algorithms, termed simultaneous generalized hill climbing (SGHC) algorithms, offer a new approach that allows practitioners to make a single algorithm run over a set of fundamentally related discrete optimization problems.

This dissertation develops a metric between elements in a set of fundamentally related discrete optimization problems (Vaughan et al. 2000). This metric is a tool for evaluating if it is advantageous to address a fundamentally related set of discrete optimization problems with a SGHC algorithm, or apply GHC algorithms to each problem in the set individually. The SGHC algorithm probabilistically moves between discrete optimization problems according to a problem generation probability function. This dissertation shows that the problem generation probability function is a stochastic process that satisfies the Markov property. Therefore, for a SGHC algorithm, movement between discrete optimization problems can be modeled as a Markov chain. Sufficient conditions are obtained that guarantee that this Markov chain has a uniform stationary probability distribution.

This dissertation presents several results regarding the performance of a SGHC algorithm where the problem generation probability function satisfies Criteria A (see Chapter 7). In particular, a lower bound for the expected number of outer loop iterations that a SGHC algorithm that satisfies Criteria A remains in a particular discrete optimization problem is presented. Additionally, a lower bound is obtained for the probability that a SGHC algorithm that satisfies Criteria A will continue to execute over the solution space of a particular discrete optimization problem. Both lower bounds are shown to be functions of the problem generation probability function. Additionally, sufficient conditions are presented that guarantee that a SGHC algorithm will visit the globally optimal solution over all the discrete optimization problems in a set of fundamentally related discrete optimization problems.

This dissertation contains computational results for an Air Force manufacturing problem and an instance of the multiple traveling salesman problem (MTSP) that validate the usefulness of simultaneously addressing a set of discrete optimization problems using GHC. The computational results for the MTSP suggest that the SGHC algorithm outperforms the GHC algorithm, as measured by the means of the optimal tour distances across multiple replications.

The research presented in this dissertation suggests several new directions of study. For example, addressing the MTSP using a SGHC algorithm with a variety of neighborhood functions will be studied (i.e., the city exchange neighborhood function). Moreover, the efficiency of approaching a set of discrete optimization problems where the individual discrete optimization problems are variants of the TSP will be explored (i.e., Geometric, Bottleneck, Rural Postman).

The SGHC algorithm can also be used as a tool for evaluating the efficiency of neighborhood functions for local search algorithms used to approach a set of fundamentally related discrete optimization problems. To evaluate the efficiency of a set of $N \in Z^+$ possible neighborhood functions for a particular discrete optimization problem, each neighborhood function coupled with the discrete optimization problem can be considered as a separate discrete optimization

problem, forming a set of N fundamentally related discrete optimization problems. This set of fundamentally related discrete optimization problems will be addressed with a SGHC algorithm, in the hopes of identifying the optimal neighborhood function.

The SGHC algorithm is a new approach for addressing a set of fundamentally related discrete optimization problems that can be more efficient than the traditional approach of addressing each discrete optimization problem in the set S individually with a local search algorithm. For example, SGHC algorithms allow practitioners to make a single algorithm run over a set of fundamentally related discrete optimization problems. Moreover, the convergence results presented in this dissertation imply that whenever the underlying GHC algorithm converges to a globally optimal solution for each discrete optimization problem in S, a SGHC algorithm can be developed to address S that is guaranteed to visit the globally optimal solution over the set of fundamentally related discrete optimization problems. Therefore, a SGHC algorithm can be implemented instead of addressing each discrete optimization problem in S individually with a local search algorithm without losing any convergence properties. Moreover, the computational results presented suggest that a SGHC algorithm can outperform the GHC algorithm. The development of the SGHC algorithm and the mathematical results in this dissertation make it possible for the SGHC algorithm to be adapted and used to approach a variety of real world problems.

Bibliography

Aarts, E., Lenstra, J.K., 1997, *Local Search in Combinatorial Optimization*, Wiley and Sons, New York, New York.

Aarts, E., Korst, J., 1989, *Simulated Annealing and Boltzmann Machines A Stochastic Approach to Combinatorial Optimization and Neural Computing*, John Wiley and Sons, Chichester.

Anily, S., and Federgruen, A., 1987, "Simulated Annealing Methods with General Acceptance Probabilities", *Journal of Applied Probability*, 24, 657-667

Atkinson, K.E., 1989, *An Introduction to Numerical Analysis*, John Wiley and Sons, New York, New York.

Bock, F., 1958a., "An Algorithm for Solving 'Traveling-Salesman' and related network optimization problems": Abstract. Bulletin Fourteenth National Meeting of the Operations Research Society of America, 897.

Bock, F., 1958b, "An Algorithm for Solving 'Traveling-Salesman' and related network optimization problems", Manuscript associated with talk presented at the Fourteenth National Meeting of the Operations Research Society of America.

Cerny, V., 1985, "Thermodynamical Approach to the Traveling Salesman Problem: an Efficient Simulation Algorithm", *Journal of Optimization Theory and Applications*, 45, 41-51.

Croes, G.A., 1958, “A Method for Solving Traveling Salesman Problems”, *Operations Research*, 791-812.

Dueck, G. and T. Scheuer, 1990, “Threshold Accepting: A General Purpose Optimization Algorithm Appearing Superior to Simulated Annealing”, *Journal of Computational Physics*, 90, 1616-175.

Eglese, R.W., 1990, "Simulated Annealing: A Tool for Operational Research", *European Journal of Operational Research*, 46, 271-281.

Fischer, C.E., Gunasekera, J.S., Malas, J.C., 1997, “Process Model Development for Optimization of Forged Disk Manufacturing Processes”, *Steel Forgings*, Second Volume, ASTM STP 1257, E.G. Nisbett and A.S. Melilli, Editors, American Society for Testing and Materials.

Fleischer, M.A., 1995, “Simulated Annealing: Past, Present, and Future”, *Proceedings of the 1995 Winter Simulation Conference*, 155-161.

Garey, M.R., Johnson, D.S., 1979, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, New York, New York.

Glover, F., and Laguna, M., 1997, *Tabu Search*, Kluwer Academic Publishing, Norwell, Massachusetts.

Gunasekera, J.S., Fischer, C.E., Malas, J.C., Mullins, W.M., Yang, M.S., 1996, “Development of Process Models for Use with Global Optimization of a Manufacturing System”, Proceedings of the ASME Symposium on Modeling, Simulation and Control of Metal Processing, ASME-international Mechanical Engineering Congress, Atlanta, GA, November, 1996.

Hajek, B., 1988, "Cooling Schedules for Optimal Annealing", *Mathematics of Operations Research*, 13, 311-329.

Hillier, F. and Lieberman, G., 1995, *Introduction to Operations Research*, McGraw-Hill, New York, New York.

Isaacson, D. and Madsen, R., 1976, *Markov Chains Theory and Applications*, Robert E. Krieger Publishing Company, Inc., Malabar, Florida.

Jacobson, S.H., Sullivan, K.A., Johnson, A.W., 1998, "Discrete Manufacturing Process Design Optimization Using Computer Simulation and Generalized Hill Climbing Algorithms", *Engineering Optimization*, 31, 247-260.

Jacobson, S.H., Yucesan, E., 2000, "A Generalized Hill Climbing Algorithm Framework for Studying the Performance and Convergence of Simulated Annealing and Local Search Algorithms", Technical Report, University of Illinois at Urbana-Champaign, Urbana, Il., (submitted for publication).

Johnson, A.W., Jacobson, S.H., 2000a, "A Class of Convergent Generalized Hill Climbing Algorithms", To Appear in *Applied Mathematics and Computation*.

Johnson, A.W., Jacobson, S.H., 2000b, "On the Convergence of Generalized Hill Climbing Algorithms", To Appear in *Discrete Applied Mathematics.*

Kirkpatrick, S., Gelatt, Jr., C.D. and Vecchi, M.P., 1982, "Optimization by Simulated Annealing", *IBM Research Report* RC 9355.

Kirkpatrick, S., Gelatt, Jr., C.D. and Vecchi, M.P., 1983, "Optimization by Simulated Annealing", *Science* 220, 671-680.

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

Liepins, G.E., Hilliard, M.R., 1989, "Genetic Algorithms: Foundations and Applications", *Annals of Operations Research*, 21, 31-58.

Lin, S., 1965, "Computer Solutions of the Traveling Salesman Problem", *Bell Technical Journal*, 44, 2245-2269.

Mitra, K., F. Romeo, and A. L. Sangiovanni-Vincentelli (1986) "Convergence and Finite-time Behavior of Simulated Annealing", *Advances in Applied Probability*, 18, 747-771.

Reiter, S, Sherman, G., "Discrete optimizing", *Journal of the Society for Industrial and Applied Mathematics* 13, 864-899.

Schuur, P.C., 1997, "Classification of Acceptance Criteria for the Simulated Annealing Algorithm", *Mathematics of Operations Research*, 22(2): 266-275.

Tovey, C.A., 1983, "On the Number of Iterations of Local Improvement Algorithms", *Operations Research Letters*, 2, 231-238.

Vaughan, D., Jacobson, S.H., Armstrong, D., 2000, "A New Neighborhood Function for Discrete Manufacturing Process Design Optimization using Generalized Hill Climbing Algorithms", ASME *Journal of Mechanical Design*, 122 (2), 164-171.