RECLAMATION

Managing Water in the West

Technical Report S&T-2013-3906

Phase 2 – Advanced Algorithms for Hydropower Optimization

U.S. Department of the Interior
Bureau of Reclamation
Technical Service Center
Denver, Colorado

November 2013

Legal Notice

Cover photo: An aerial view of Glen Canyon Dam (Reclamation file photo).

Technical Report S&T- 2013-3906

Phase 2 –Advanced Algorithms for Hydropower Optimization

This Research Report was prepared by the U.S. Department of the Interior, Bureau of Reclamation, Technical Service Center, Denver, Colorado.

David Harpman — 19 Nov 2013

Prepared by: David A. Harpman, Ph.D.
Natural Resource Economist
Economics, Planning and Technical Communications,
86-68270

Date

Todd Gaston — 11/19/2013

Peer review: Todd L. Gaston, MSc.
Natural Resource Economist
Economics, Planning and Technical Communications,
86-68270

Date

Toby Steves — 11-19-2013

Peer Review: Toby L. Steves, PE
Electrical Engineer
Infrastructure Services
86-68450

Date

Hoover Dam

Celebration of Reclamation's 100th Anniversary at Hoover Dam.

Acknowledgments

This document has benefited immeasurably from the gracious assistance and technical guidance provided by the collaborative research team. In alphabetical order, the members of the team are:

Craig A. Bond, Colorado State University
Darrel G. Fontane, Colorado State University
John B. Loomis, Colorado State University
Mr. Thomas D. Veselka, Argonne National Laboratory

Any errors are the sole responsibility of the author.

Project Funding

This research project was funded in Fiscal Years 2012 and 2013 by the U.S. Bureau of Reclamation's Science and Technology Program, Project Identification Number 3906. Additional support from the University of Denver, Colorado State University and Argonne National Laboratory is gratefully acknowledged.

Credits

Dr. David Raff, Hydraulic Engineer, with the U.S. Army Corps of Engineers, Institute for Water Resources, Alexandria, Virginia, constructed the graphics shown in Appendix 4 using MATLAB version 4.x. His technical assistance is gratefully acknowledged.

Contents

Page

Tables

Table Page

Figures

Figure Page

Executive Summary

The invention of microcomputers (circa the mid-1980s) started an era of unparalleled research in the field of heuristic optimization algorithms. These new algorithms are based on physical and biological processes rather than calculus-based approaches. Four promising evolutionary algorithms (EA's) were identified from the emerging literature; the real coded genetic algorithm (RCGA), differential evolution (DE), particle swarm optimization (PSO) and artificial bee colony optimization (ABCO). These four EAs were applied to an important hydropower problem—the hydropower unit dispatch problem. The goal of the unit dispatch problem is to select which generation units to operate and their output levels to meet generation and ancillary service needs, while using the least amount of water. A suite of replicated experiments using these four EAs was undertaken on the unit dispatch test problems. These experiments explored the influence of initialization approaches, convergence criteria, dimensions of the problem, and the role of binding constraints. The aggregate experimental evidence indicates these algorithms can reliably solve the unit dispatch problem, within acceptable time-frames. Experimental results suggest for the unit dispatch problem, the choice of initialization approach has no effect on solution times. Replicated experiments indicate intelligent stopping rules, which monitor convergence progress, dominate uninformed approaches. For DE, PSO and ABCO, convergence times become longer as the problem size is increased and for DE and PSO, when some constraints are binding. For RCGA and ABCO convergence speeds may improve in the presence of binding rough zone constraints. A large number of algorithm specific experiments were conducted to inform these algorithm performance comparisons and a subset of these are described in this report. The reliability of the EA's examined proved to be excellent and their solution speeds are fast enough for use in operational decision-making. The hydropower unit dispatch problem is nonlinear, non-convex and discontinuous. These conditions preclude the application of traditional calculus based algorithms. Evolutionary algorithms are more robust under these conditions. They could provide near real-time solutions and guidance for everyday operational decisions at Reclamation's hydropower plants. Continuing development and testing of these heuristic algorithms, leading towards operational deployment, is advised.

Introduction

Within the last 30 years, a variety of new optimization heuristics have been described in the power engineering literature. These heuristic approaches rely on innovative search techniques, drawn from biological and physical processes.

Although computationally intensive, these methods can solve difficult constrained optimization problems, like the unit dispatch problem, quickly and reliably.

This research describes several of the most promising of these new optimization approaches, applies them to example test dispatch problems and systematically assesses their performance. The goal of this effort is to identify algorithms which can help guide hydropower unit dispatch decisions and improve efficiency, generating more electric power with less water.

Reclamation plays a highly visible leadership role in the electric power industry. The Bureau is the second largest producer of hydroelectric power in the United States and, if it were a utility, would rank as the ninth largest electric utility on the basis of production capacity. We operate 58 power plants with an installed capacity of 14,809 MW and produce approximately 40 billion kWh of energy annually. As a registered generation and transmission owner-operator, Reclamation plays a key role in regional reserve sharing agreements and is a member of the Western Electricity Coordinating Council (WECC).

By statute, Reclamation's electric power must be marketed at the lowest possible rates consistent with sound business practice. The goal of this research project is to identify and apply advanced approaches allowing the operation of Reclamation hydropower plants in a more efficient manner generating more electricity per acre-foot of water released. This research is fully consistent with Reclamation's mandate and reflects our stewardship responsibilities as a water and power provider. Improved efficiency will result in the generation of more electric power using less water benefitting water and power users, as well as the American taxpayer.

Overview of Phase 1 Research

The focus of Phase 1 of this research effort was to apply selected evolutionary algorithms (EAs) to the dynamic constrained hydropower dispatch problem. In Phase 1 of this research project (Harpman 2012), three promising evolutionary algorithms were identified from the emerging heuristic optimization literature; the real coded genetic algorithm (RCGA), differential evolution (DE) and particle swarm optimization (PSO). A relatively extensive suite of replicated experiments were conducted to assess the performance characteristics of these three algorithms. These experiments systematically explored the influence of initialization approaches, convergence criteria, the dimensions of the problem, the role of problem inputs and the effects of binding constraints. The results show the convergence behavior of evolutionary algorithms differs from traditional calculus based approaches. Relative to the calculus based approaches, evolutionary algorithms exhibit longer solution times—characterized by rapid identification of the region containing the optimum, with relatively slow local convergence. The

choice of different initialization approaches appears to have no effect on solution times, for the particular problem examined. Replicated experiments indicate convergence times for all three of the EAs examined are longer for higher dimension problems. For DE and PSO, convergence times increase when additional constraints are binding. Input price vectors with greater dynamic ranges appear to degrade convergence times for DE, with mixed results for PSO. The aggregate experimental evidence indicates these algorithms can reliably solve this class of problem, within acceptable time-frames.

Focus of Phase 2 Research

The focus of Phase 2 of this research effort was the application of selected evolutionary algorithms to the hydropower unit dispatch problem. Phase 2 of this research project builds upon the earlier Phase 1 effort but is focused on hydropower unit dispatch, a Type 2 optimization problem, which is described subsequently.

The strength of evolutionary algorithms is they are able to successfully solve a broad range of problems, including discrete, discontinuous and non-convex problems. The hydropower unit dispatch problem is such a problem, and one with widespread, practical, everyday management application at Reclamation hydropower plants.

Types of Hydropower Optimization Problems

Five different types of hydropower optimization problems are discussed in the literature. Following the descriptions provided by Curtis, Parker and Scoggins (2012), these are reported in Table 1.

Type 1 optimization is the optimization of one single generating unit. The real power output of the unit is maximized at a given reservoir elevation by identifying the release level which produces the most energy per unit of water released.

Type 2 optimization is the focus of this report. The goal of Type 2 optimization is to simultaneously identify which units will operate and the output levels of all of the selected generator units in the powerhouse in order to meet the required level of generation, reserves and ancillary services while using the least amount of water.

The Type 2 optimization problem is complex and difficult to solve because each turbine in the powerhouse may have different performance characteristics. This is most evident for different size units and units with different designs and manufacturers. Some authors (Curtis, Parker and Scoggins 2012) have reported that even among units with the same manufacturer and identical designs, there can be small differences in the production process. They point out that differential wear, which can result from differences in unit run-times may also give rise to subtle differences in unit performance characteristics. Performance differences can also result from non-uniform maintenance and repair procedures.

In recent years, improved professional standards (for example, see International Electrotechnical Commission 1999) in tandem with modern design and production techniques have resulted in a high degree of uniformity across turbines of the same design, produced by the same manufacturer. Reclamation's experience (Hulse 2013) suggests that turbines of the same design produced by the same manufacturer typically have the same performance characteristics, within the limits of the unit performance test error, as described in the ASCME guidelines (American Society of Mechanical Engineers 2011).

The specifics of the unit dispatch problem, a Type 2 optimization, are described more fully and more formally in subsequent sections of this document.

Table 1.—Optimization Classification Scheme

Optimization Problem	Definition
Type 1	Optimization of the output of a single turbine
Type 2	Optimal joint operation of all of the turbines in the powerhouse
Type 3	Optimal operation of all hydropower stations within a river basin.
Type 4	Optimal operation of the hydropower stations across multiple river basins in a geographic region
Type 5	Optimal coordination of all generating resources (hydro, thermal, renewable) within a geographic region.

Type 3 optimization problems are designed to identify the optimal operation of all of the hydropower plants within a river basin. This problem must necessarily consider not only the physical and engineering characteristics of each plant individually, but the interlinked hydrologic impacts of each plant on the other plants in the system.

Type 4 optimization problems encompass the joint optimal operation of all of the hydropower plants in a geographic region.

Finally, Type 5 optimization problems are focused on the coordinated optimal operation of all generating resources within the geographic (usually interconnected) region of interest. These kinds of optimization problems may or may not explicitly characterize the transmission system, emissions targets and other constraints pertinent to the operation of the interconnected electric power system, as a whole.

The Type 2 Unit Dispatch Problem

The unit commitment or unit dispatch problem is a complex 2-step mathematical optimization problem. Solution of this problem requires (a) identification of the combination of available hydropower units to operate (or shut down) in a single time-step, such as a 1–hour period, and, (b) how to managed the committed hydropower units in an optimal manner. The decision to operate or shut down a unit is a binary (0/1) decision, typically with some associated cost and often with some minimum time constraint imposed between startup and shutdown decisions. Assuming there are n available hydropower units, the unit dispatch problem is of size 2^n. This aspect of the problem poses a potentially large and difficult-to-solve integer programming effort. Once the optimal units have been committed, the economic dispatch problem, described previously in Phase 1 of this project, is solved for those units.

Understanding Type 2 Optimization

The goal of Type 2 optimization is to simultaneously identify which generator units at the plant will operate and the output levels of all the selected generator units in order to meet required levels of generation, reserves and ancillary services, while using the least amount of water.

Unit Characteristics

At the heart of the Type 2 optimization problem is the important insight that each generator unit/turbine combination in the powerhouse may be different. Most obviously, the sizes, capacities and designs of some generator units may differ from other units at that plant. For example, some units may have a nameplate capacity of 50 MW while others are rated for 65 MW. Similarly, some units may be designed to operate most efficiently at different reservoir elevations (e.g. higher or lower reservoir levels) than others. In these cases, there are clear reasons for the differences in performance characteristics between units. Figure 1 illustrates the performance characteristics for 24 units with three different

maximum generation capabilities. Units 1-9 and 22-24 are rated at 143.75 MVA. Units 19-21 are rated at 707.8 MVA and Units 22-24 are the largest units, rated at 825.64 MVA. As shown, the smaller units achieve their highest efficiency at lower output levels.

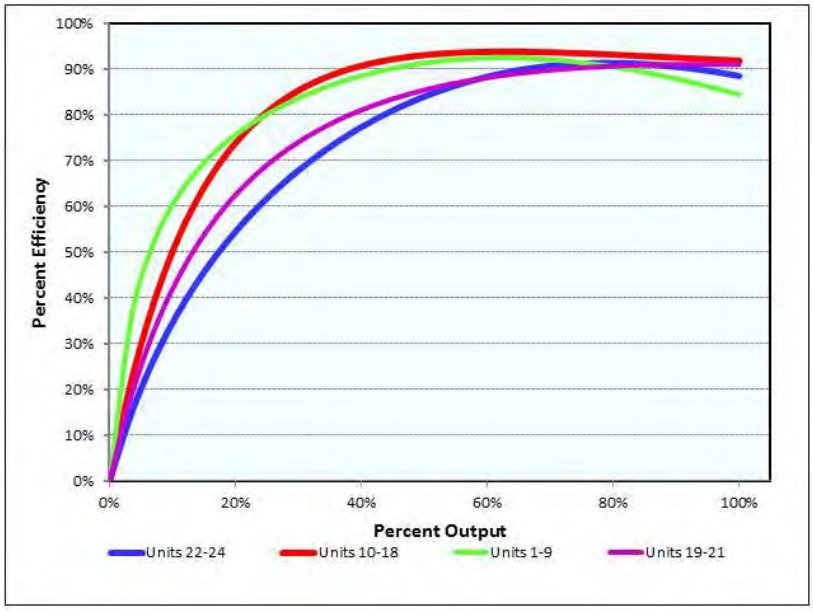

Figure 1.—Grand Coulee Unit Performance Characteristics.

Powerplant and power system operational considerations also play a large role in Type 2 optimization. A short discussion of these other considerations is presented below.

Available Units

Although there may be 10 generator units at a particular powerplant, it is not uncommon for one or more of these units to be offline and therefore unavailable for dispatch or use. Units are routinely taken offline and idled for scheduled maintenance, unscheduled maintenance, repairs and other reasons. Depending on the particulars of the situation, these offline periods can last a few hours, a few days, or a number of months.

Offline units are unavailable for dispatch. Any Type 2 optimal dispatch solution must account for offline units and dispatch the other available (online) units, to meet electrical load in the most efficient fashion.

Must Run Units

In some cases, particular generation units at a powerplant are designated as "must run" or always run units. If the powerplant is in operation, these units are dispatched fully or partially. Must run units are designated for a variety of logical, operational and environmental reasons. For example, one unit may be reserved for "station service" to provide electric power for operating the powerplant and control equipment at the plant. Alternatively, one unit may be in continuous "must run" status in order to provide power system support or transmission support. At other plants, a unit may be designated as a must run unit to ensure the minimum downstream flow requirements are met.

By definition, must run units are always dispatched. A Type 2 optimal dispatch solution must account for these continuously operating units and dispatch the other available units to meet the remaining electrical load.

Preferred Dispatch Order

At many powerplants there are formal and informal preferences for the order in which generator units are dispatched. This preferred dispatch order may extend to some or all of the units at the powerhouse. Depending on the plant and the powerplant operator on duty, generator units may be dispatched in a particular order so as to balance the use and wear across units, to avoid noise or vibration in the powerhouse, or for other operator specific reasons which may or may not be intuitively obvious.

At powerplants where a preferred dispatch order is observed, a Type 2 optimal dispatch solution must account for and accommodate this preference ordering. This additional operating constraint may or may not have an effect on the overall efficiency of the optimal solution.

Spinning Reserve

Spinning reserves are defined as a designated block of unloaded generating, connected to an output bus, synchronized to the electric system, and ready to take immediate load. When a generator supplies spinning reserve services, it will increase output in response to an outage situation. When a unit is fully or partially designated to spinning reserve, its output level is reduced so it can meet the spinning reserve obligation without exceeding the maximum capability of the generator.

Units, or portions of units, designated to meet spinning reserve requirements are unavailable for real power dispatch. Any Type 2 optimal dispatch solution must account for units, or portions of units, on spinning reserve and dispatch only the

unloaded portion of the unit or other available (online) units, to meet electrical load in the most efficient fashion.

Nonspinning Reserve

Nonspinning reserves are defined as generating capacity that is unloaded, connected to an output bus, and not synchronized to the electricity system. Depending on balancing area regulations, these resources must be capable of being brought online in 10 minutes if it is offline, and which is capable being operated for at least two hours. When a hydropower unit is designated for nonspinning reserves it is often dewatered, idle and cannot be dispatched to meet load.

Units designated to meet nonspinning reserve requirements are unavailable for power dispatch (they are on reserve). Any Type 2 optimal dispatch solution must account for units on nonspinning reserve and dispatch the other available (online) units, to meet electrical load in the most efficient fashion.

Regulation

Regulation is the amount of operating reserve capacity required by the control area to respond to automatic generation control to ensure the Area Control Error (ACE) remains within the performance standards described in North American Electricity Reliability Council (NERC) (2011). ACE is the instantaneous difference between a Balancing Authority's net actual and scheduled interchange, taking into account the effects of frequency bias and correction for meter error.

Units designated for providing regulation down services must operate at sufficiently high output levels such that sudden decreases in load will not reduce generation below their technical or regulatory minimum output levels. To provide regulation-up services, unit generation levels must be sufficiently low such that a power plant can respond to instantaneous increases in grid loads without exceeding their output capability.

Units designated to meet regulation (up and down) requirements are available only for limited power dispatch. Any Type 2 optimal dispatch solution must account for these limited operating ranges and dispatch the other available (online) units, as necessary, to meet electrical load in the most efficient fashion.

Condensing/Motoring

Depending on the design of the plant, hydropower units can be operated as synchronous condensers to increase or decrease reactive power on the interconnected electricity grid. In this mode (also referred to as "motoring"), the

hydro unit's breaker switch is closed and no water is released to drive the turbine. Air is pumped into the turbine to allow it to spin freely. Real electric power is supplied to the unit and the unit's generator acts like a motor, rotating the turbine shaft and absorbing reactive power from the system. Depending on the control settings, condensing units can also be used for voltage and frequency control in the system.

In general, condensing/motoring units are unavailable for real power dispatch. Depending on the market rules however, these units may sometimes be counted towards the spinning reserve requirements. Any Type 2 optimal dispatch solution must account for motoring units and dispatch the other available (online) units, to meet electrical load in the most efficient fashion.

Prohibited Operating Zones

Prohibited operating zones are production, output, or operational regions which create excessive vibrations of the plant equipment (also known as, 'rough zones') or output zones which might result in hydraulic cavitation. Many generator/turbine units have rough zones, sometimes more than one. These prohibited operating zones may vary with head, further complicating this problem. Units of different designs and wear-status typically have rough zones in different portions of their output surfaces. Sustained operation of a unit within a rough zone is not permitted under normal operating conditions. It is also advisable to change output levels quickly and move through prohibited operating zones as rapidly as feasible.

Prohibited operating zones place additional constraints on the output levels of generator/turbine units. Continuous generation within these zones is not allowed under ordinary circumstances. Movement of the set point from below the prohibited operation zone to an output level above the prohibited operating zone will necessarily require a rapid transition through it. Any Type 2 optimal dispatch solution must account for these prohibited operating zones and dispatch the unit in a manner which avoids these regions, or else dispatch the other available (online) units, to meet electrical load in the most efficient fashion

Start/Stop Costs

Hydropower generation units which are brought online from an offline state, or taken offline from an online status, incur some additional amount of equipment wear. Additionally, there is some amount of labor and time required to implement these status changes. Although empirical evidence is limited, at least conceptually, some costs are incurred for unit start ups and unit stops. Although some unit start ups and stops are necessary to the coordinated operation of any powerplant, it is advantageous to limit these, to the extent practical.

Any Type 2 optimal dispatch solution must account for start up and shut down costs, in some fashion. Note that consideration of this constraint requires a dynamic, or period to period, accounting of unit activities. Ideally, a prudent dispatch solution would avoid unnecessary unit cycling while dispatching units in the most efficient fashion.

Minimum Up/Down-Times

Explicit consideration of minimum up and down times is often related to start and stop costs and sometimes engineering concerns. As applied, for some or all units, there is a minimum period of time that a unit can be offline, once it has been shut down and a minimum period of time a unit can be run, when it is started up. These constraints may be imposed to prevent excessive unit cycling, to enforce cooling requirements, reduce unit wear and prevent excessive start and stop costs. Where present, a Type 2 optimal dispatch solution must account for these minimum run and minimum down time constraints. Note that consideration of these constraints requires a dynamic, or period to period, accounting of unit states. Ideally, a prudent dispatch solution would satisfy these constraints while dispatching all available units in the most efficient fashion.

Unit Leakage

When units are offline and the head gate to the unit is closed, there is little or no leakage of water through the unit. When units are online and available, but not generating, some amount of leakage through the wicket gates can be expected. While the amount of leakage is generally small, as a percentage of the total release capacity, it can vary to a greater or lesser degree across units depending on a variety of factors.

Generator/turbine units which are either spinning or motoring/condensing will typically pass some water through the wicket gates without generating any real power. Water which leaks through the wicket gates is effectively unavailable for use, or wasted. Depending on the age, condition and wear-status of the available units, they may leak differentially to a greater or lesser extent. Although some degree of leakage is unavoidable, it is clearly desirable to minimize the amount of leakage which takes place.

Any Type 2 optimal dispatch solution must account for potential leakage. All other factors being the same, units which leak *more* should be dispatched first and not reserved for motoring/condensing operations, while the other available (online) units should be dispatched, to meet electrical load in the most efficient fashion.

Reclamation Generation Units

As of May 2012, Reclamation owns and operates a total of 57 hydropower plants. These range in size from the Lewiston powerplant with an installed capacity of 0.350 MW to Grand Coulee powerplant with an installed capacity of 6,809 MW. Depending on the powerplant, there may be a single generator unit, or there may be two or more generator units. Fifteen (26.32%) of the 57 powerplants in the Reclamation inventory are single generation unit plants. Nineteen (33.33%) of the 57 powerplants have 2-generation units. Figure 2 illustrates the percentage of Reclamation power plants by number of generation unit.

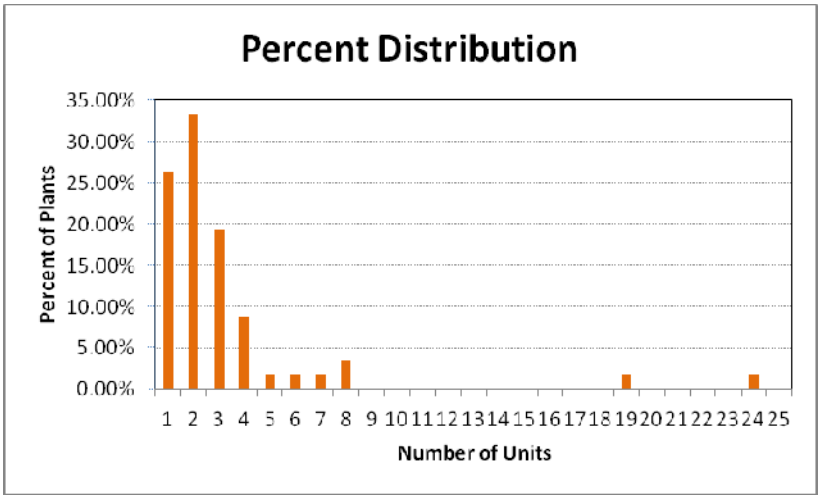

Figure 2.—Units at Reclamation Powerplants.

As shown in this graph, the majority of Reclamation powerplants (50, or 87.72%) have four or fewer generation units. Approximately, 12% of Reclamation's powerplants have more than four generation units. Among the plants with the largest number of generation units are; Glen Canyon (8-units), San Luis (8-units), Hoover (19-units) and Grand Coulee (24-units).

Unit Dispatch Test Problems

For purposes of this research effort, four unit dispatch test problems were constructed. These consisted of two, 2-unit test problems and two, 4-unit test problems. These dispatch test problems encompass small and large generator units, with and without the presence of prohibited operating zones. In broad terms, these four test problems span the range of generation facilities, and operational problems, commonly encountered in the Reclamation hydropower system.

Mathematically, these dispatch test problems are relatively simple unit dispatch problems with low dimensionality. More importantly, they can be solved successfully using total enumeration, the Excel solver, nonlinear programming and other available techniques. They are small, compact and relatively tractable problems. By design, they facilitate detailed exploration of the efficacy and performance of the evolutionary algorithms examined here.

The characteristics of these four dispatch test problems are summarized in Table 2. A detailed exposition of these problems can be found in Appendix 6 and the specifications for the storage reservoir and critical elevations can be found in Appendix 5. As shown, there are two generator units in Test Problem 1 and there are two generator units in Test Problem 2. In both test problems, there is one small generation unit, with a nominal capacity of 64 MW, and a large generation unit, with a nominal capacity of 153 MW.

Table 2.—Summary of Test Problems

Label	Unit	Capacity (MW)	Efficiency	Features
Test Problem 1	1	63.7378[1]	0.90	
	2	153.4053[2]	0.85	
Test Problem 2	1	63.7378	0.90	Rough zones
	2	153.4053	0.85	Rough zones
Test Problem 3	1	63.7378	0.90	
	2	153.4053	0.85	
	3	63.7378	0.90	
	4	153.4053	0.85	
Test Problem 4	1	63.7378	0.90	
	2	153.4053	0.85	
	3	63.7378	0.90	Rough zones
	4	153.4053	0.85	Rough zones

In Test Problem 1, there are no prohibited operating zones for either generator unit. For Test Problem 2, both the small and the large generation unit have prohibited operating zones. Figure 3 illustrates the plan (top) view schematic common to both Test Problem 1 and Test Problem 2.

[1] The maximum occurs at an elevation of 2008.19 and a release of 3000 cfs.
[2] The maximum occurs at an elevation of 2008.19 and a release of 9000 cfs.

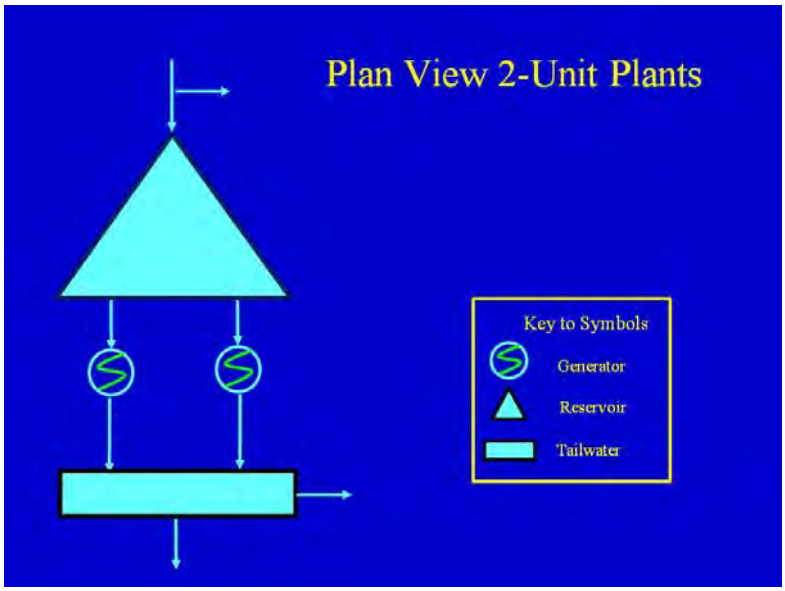

Figure 3.—Plan View of Test Problems 1 and 2.

As shown in Table 2, there are four generation units in Test Problem 3 and four generation units in Test Problem 4. In each of these test problems, there are two small generation units, with a nominal capacity of 64 MW, and two large generation units, with a nominal capacity of 153 MW. In Test Problem 3, none of the four units have prohibited operating zones. For Test Problem 2, one of the small units and one of the large generation units have prohibited operating zones. Figure 4 illustrates the plan (top) view schematic common to both Test Problem 3 and Test Problem 4.

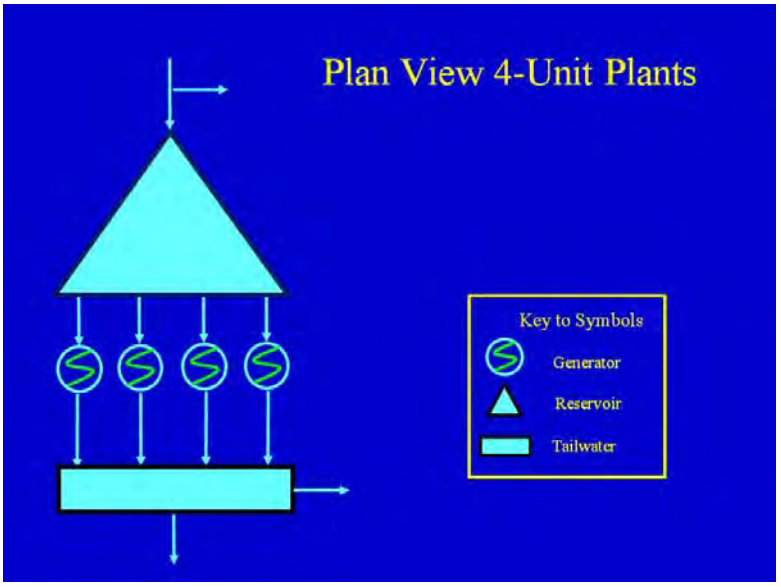

Figure 4.—Plan View of Test Problems 3 and 4.

Problem Representativeness

The unit dispatch model developed in this research project is applied to the 2-unit and 4-unit dispatch problems, as described in an earlier section of this document. Admittedly, this research application does not encompass the full range of units represented at all Reclamation projects. It should be recognized however, on the basis of the number of units, the 2-unit test problems characterize 59.65% of Reclamation plants and the 4-unit test problems are representative of 87.72% of all current Reclamation hydropower plants. The dispatch test problems developed for this research project provide an adequate representation of existing Reclamation powerplants.

Problem Limitations

The 2-unit and 4-unit dispatch problems employed in this research project do not characterize all of the potential conditions which must be considered by plant operators. The preceding section of this document entitled, "Understanding Type 2 Optimization," provided some insight into the complexity of the dispatch decision. Figure 5 summarizes the majority of these concepts and provides a useful overview of the factors explicitly characterized in the test problems, and those which were not.

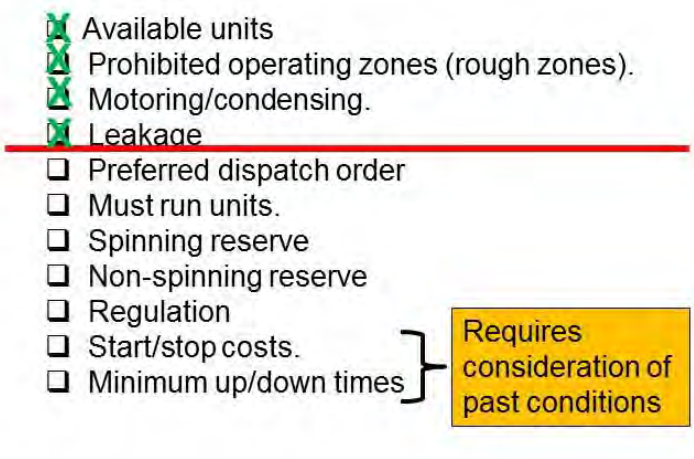

Figure 5.—Test Problem Limitations.

The unit dispatch test problems developed for this research effort encompass the plant conditions shown above the red line in Figure 5. All of the software programs described in this document can characterize the number of available units, prohibited operating zones, designated motoring/condensing units and

reflect unit leakage. These conditions can be varied by the user and are reflected in the optimization results returned by the algorithms.

Since there were limits on the resources available for this research, there were necessarily some trade-offs which had to be made between functionality and the research objectives. Consequently, the unit dispatch problems employed here do not characterize all of the possible plant conditions faced by dispatchers. The conditions below the red line in Figure 5 are not characterized in the unit test problems used in this research project. The majority of the conditions below the red line, including the preferred dispatch order, must run units, spinning reserve, non-spinning reserve and regulation would be relatively easy to accommodate in the existing test problem framework. The final two conditions; start/stop costs and minimum up/down times, would require a more complex problem specification which considers the state of each unit in past time periods.

For simplicity reasons, the efficiency parameter for each generator unit is a scalar constant over the range of release and head. In general however, the efficiency of a Francis turbine varies, depending on the head and the release rate. Furthermore, this relationship is unique to the design of each turbine runner and the site where it is installed. A further explanation of this topic can be found in Harpman (2012 Appendix 4).

Selected Terms

Like any branch of science, there are some terms used to describe mathematical optimization approaches which are not commonly encountered in other fields. As an aid to understanding the narrative which follows, it will be useful to define some of these terms.

Algorithm

"A detailed sequence of actions to perform to accomplish some task. Named after an Iranian mathematician, Al-Khawarizmi. Technically, an algorithm must reach a result after a finite number of steps, thus ruling out brute force search methods for certain problems, though some might claim that brute force search was also a valid (generic) algorithm. The term is also used loosely for any sequence of actions (which may or may not terminate)" (Computer Dictionary Online 2010).

Heuristic

"A rule of thumb, simplification, or educated guess that reduces or limits the search for solutions in domains that are difficult and poorly understood. Unlike (true) algorithms, heuristics do not guarantee optimal, or even feasible, solutions

and are often used with no theoretical guarantee" (Computer Dictionary Online 2010).

In practice, the term algorithm is often used interchangeably with the term heuristic. However, mathematicians typically reserve their use of the word algorithm to describing optimization approaches for which there is a theoretical mathematical basis for expecting a favorable result. Typically, mathematicians employ the term heuristic to describe any of the non-traditional optimization approaches not supported by mathematical theory.

Objective Function

The object of mathematical optimization is to minimize or maximize a specified mathematical expression. This expression is known as an objective function.

Penalty

Many applied mathematical optimization problems have natural or logical constraints on the values which can be considered in the solution. For example, physical (quantity) measurements are typically non-negative.

One approach to characterizing constraints in a constrained mathematical optimization problem is to arithmetically disadvantage, or penalize, solution results which violate a constraint. This topic is discussed in much greater detail in subsequent sections of this document. A penalty function is used to compute the numerical magnitude of the disadvantage caused by one or more constraint violations. A penalty is the value returned by a penalty function.

Fitness

In cases where penalty functions are used to characterize constraint violations, a fitness function is maximized or minimized instead of an objective function. A fitness function returns the numerical value of the fitness—defined as the objective function value minus the value of the penalties (assuming a maximization problem) for constraint violations, if any.

Optimization Approaches

Taxonomy of Optimization Approaches

For purposes of this document and the discussion which follows, it will prove useful to provide some type of taxonomy or classification scheme to illustrate the relationships between these two optimization approaches. Figure 6 provides some structure for this discussion.

As shown in Figure 6, optimization approaches can be divided into traditional (calculus based) optimization algorithms and heuristic algorithms. The latter class of optimization methods may also be described as metaheuristics or heuristic optimizers, depending on the author and the source.

The focus of this research is on a sub-set of optimization methods which are classified as heuristic algorithms. Even so, comparison and understanding of these methods is facilitated by some familiarity with traditional methods and approaches.

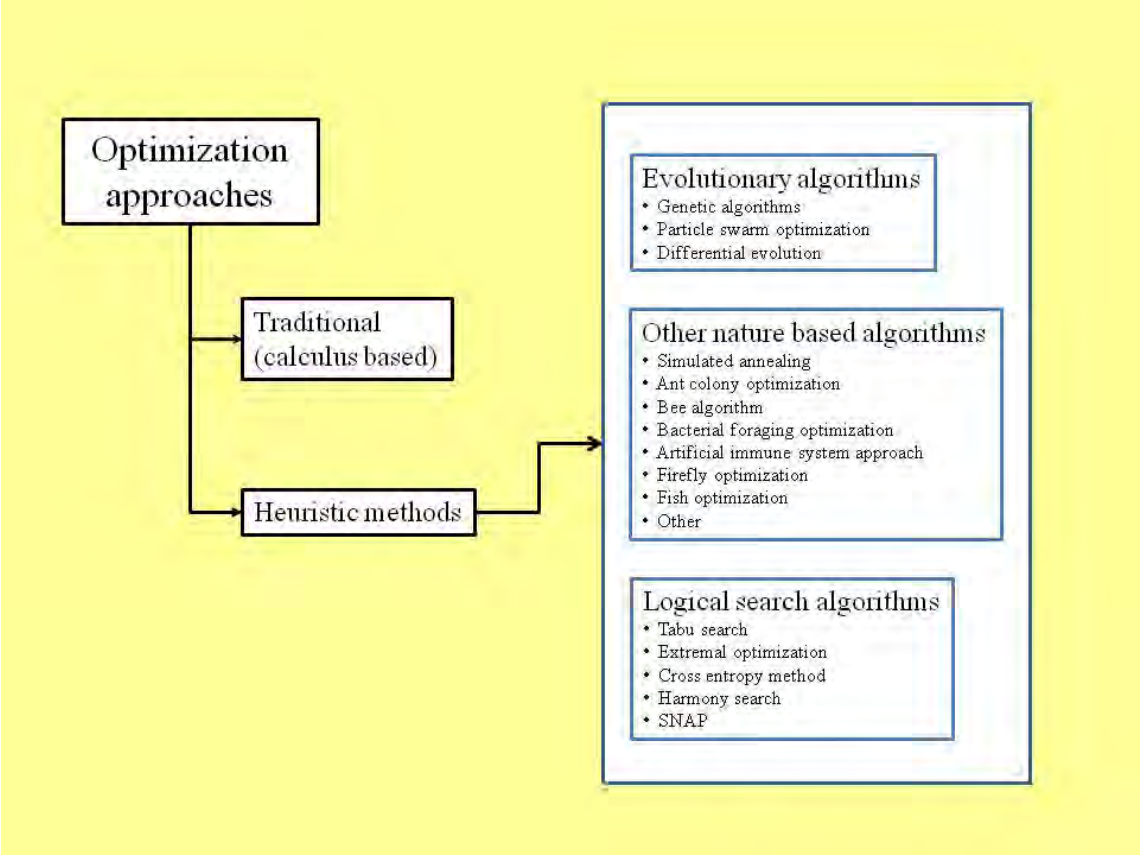

Figure 6.—Taxonomy of Optimization Approaches.

Traditional Solution Algorithms

Optimization problems have traditionally been addressed with a variety of traditional calculus based methods and throughout the remainder of this document, these approaches will be referred to as "traditional" or calculus based approaches. Calculus based optimization approaches are routinely taught to all engineers and economists. Most students of these disciplines will surely have fond memories of the many hours they devoted to mastery of this topic!

Since the time of Sir Isaac Newton (1642-1727), mathematicians, economists and engineers have collectively devoted vast amounts of effort to the study of optimization, with a particular focus on convex optimization problems with constraints. There are many books devoted to this subject, one of the many modern examples being the tome by Boyd and Vandenberghe (2006).

Numerical solution of convex optimization problems is typified by the Newton-Raphson approach and its many variants. This approach has been taught to engineers and economists since the early 1950's (for example, see Wood and Wollenberg (1996) or Rau (2003)).

As described in Press et al (1989) and Judd (1999), the Newton-Raphson approach has been largely supplanted by some of its recent and more advanced variants. At the present time, two approaches are in the forefront of current calculus based optimization technology. These are the sequential quadratic programming (SQP) method, and, the generalized reduced gradient (GRG) method. Both of these methods are aptly described in Rau (2003). The SQP method is often used in high-end commercially available optimization platforms, such as LINGO (www.lindo.com). The GRG method has found its niche as the optimization solver incorporated in all currently shipping versions of Microsoft Excel (Fylstra et al 1998). As such, it may well be the world's most frequently used optimization algorithm. In any case, it is almost certainly the most widely installed optimization package! As bundled with the ubiquitously available Excel program, the solver is broadly employed in graduate and undergraduate teaching (for example, see Weber 2007).

Heuristic Optimization Methods

The focus of this research is on the application of a subset of the heuristic optimization methods shown in Figure 6. Heuristic optimization approaches are based on the application of rules and logic which reduce the search space and allow for solution of difficult optimization problems. Generalizing rather broadly, we can classify these methods into the three categories shown; evolutionary algorithms, other nature based algorithms and logical algorithms.

Evolutionary algorithms explicitly characterize crossover, mutation and selection operators (Engelbrecht 2005). As might be expected by their name, evolutionary algorithms are based on the concept of biological evolution. These approaches are based on the improvement of an artificial population of individuals over a series of generations or iterations. Each individual carries a solution to the optimization problem. At each generation, the most fit individuals in the population reproduce and their offspring survive into the next generation, the less fit individuals die and their inferior genes are lost. The fitness of the population and the quality of the solutions found, improve over time. Genetic algorithms, differential evolution and particle swarm optimization fall into this category of algorithms.

There are an amazing variety of optimization heuristics related to living organisms, their behavior or some other natural physical phenomenon. Among these are ant colony optimization, bee optimization, firefly optimization and a host of others. As might be surmised, some of these algorithms are predicated on the collective food location strategies typified by the species.

For purposes of this document, we will classify these remaining approaches as logical heuristic search algorithms. While these may be very different from one another in search strategy, they are based on logical insights, experience and in-depth knowledge of one or more types of optimization problems. As shown in Figure 6, this category includes such well-known heuristics as Tabu search and Extremal optimization. It also includes some less well known but quite effective algorithms such as the Substitution-based Non-linear Approximation Procedure (SNAP) algorithm developed by Veselka, Schoepfle and Mahalik (2003)

Comparison of Approaches

Much of the research effort described in this report is focused on the application of evolutionary algorithms to a common hydropower optimization problem. A comparison of these two classes of algorithms and their respective suitability to this problem will provide both some background and rationale. Table 3 compares a number of pertinent characteristics of these two types of approaches.

The hydropower problems examined here are inherently nonlinear with both nonlinear and linear constraints. Both traditional and evolutionary algorithms can be applied to these types of problems. Very fast and incredibly reliable traditional algorithms are available for solving problems with linear objective functions and constraints. However, traditional algorithms are typically less efficient when applied to nonlinear objectives and nonlinear constraints. They typically require longer solution times and can fail to identify a solution more frequently in this setting.

Table 3.—Traditional and Evolutionary Algorithms

Characteristic	Traditional Algorithms	Evolutionary Algorithms
Problem formulation	Linear or nonlinear	Linear or nonlinear
Mathematical requirements	Smooth, continuous and twice differentiable	Can be piecewise, discontinuous and non-differentiable
Allowable constraints	Equality, inequality, linear or non-linear.	Equality, inequality, linear or nonlinear.
Mathematical requirements	Calculus, linear and matrix algebra operations	Primitive mathematical operators only (add, subtract, multiply, divide)
Function return	Single solution	Multiple solutions
Nature of outcome	Deterministic	Stochastic
Optimal point	Extremal point closest to starting position usually identified. This may or may not be the global optima.	Extremal point within search range usually identified. This is more likely to be the global optima.
Memory requirements	Extensive	Modest
Convergence characteristics	Slow large-scale search Fast local convergence	Fast large-scale search Slow local convergence
Solution time	Problem dependent	Often lengthy
Code implementation	Complex (very)	Unsophisticated

Many commonly encountered hydropower problems are nonlinear, nonconvex, and have discontinuities. This includes the dynamic economic dispatch problem and the unit dispatch problem examined here. Perhaps the chief strength of evolutionary programs is their applicability to these types of real-world hydropower problems, a factor which largely motivated this research effort. The mathematical requirements for applying traditional optimization algorithms are rather restrictive. Typically, traditional algorithms can only be employed when the objective function and the constraints are smooth, continuous and twice differentiable. In contrast, evolutionary algorithms can solve a much wider range of problems including those which are discontinuous, piecewise, are not convex and which cannot be differentiated.

Both traditional and evolutionary algorithms can solve constrained optimization problems with various types of constraints including equality, inequality, linear and nonlinear constraints. Traditional algorithms are less well suited to solving optimization problems with nonlinear constraints. The solution of problems with one or more equality constraints can be problematic for evolutionary algorithms.

The mathematical requirements for implementing evolutionary algorithms are far less onerous than they are for traditional (calculus based) algorithms. In both

philosophy and practice, evolutionary algorithms are not based on calculus and do not use calculus constructions for obtaining a solution. In fact, some authors consider this to be their greatest strength! Evolutionary algorithms use only primitive mathematical operators such as addition, subtraction, multiplication and division. Traditional algorithms are, of course, founded in calculus concepts. As a result, they use not only gradient vectors (vectors of first partial derivatives) and hessian matrices (matrices of second partial derivatives), but also have advanced linear algebra requirements. These advanced mathematical constructs are error prone to derive and code, difficult to implement numerically and require an extremely high degree of knowledge and skill on the part of the researcher. Judd, a master of understatement, writes "Many readers could write acceptable unconstrained optimization code, but it is much more difficult to write good, stable, reliable code for constrained optimization (Judd 1999, page 142)

Traditional (calculus based) optimization algorithms return one single solution. It is *the* solution to the problem, as every economics and engineering student is acutely aware. A fundamental difference between traditional and evolutionary algorithms is that evolutionary algorithms return a population of solutions. This difference in solution paradigm is both unfamiliar and potentially confusing.

To expand upon this concept, we must recall that evolutionary algorithms characterize a population of individuals. This population is of say size, np, which could consist of from 5 to 100 individuals or more. Fundamentally, each of these np individuals stores a solution (in some cases, more than one). The stored solution consists of not only the optimal function value, but the vector of values which produces it. As the evolutionary process proceeds, each of these np solutions evolves and becomes better, or more "fit." When the evolutionary process terminates, the result is np, not necessarily unique, individual solutions--not one single solution. As a practical matter, the analyst will often choose to report the best of these np individual solutions as *the* solution. Since evolutionary algorithms are probabilistic in nature, each new run will produce slightly different results (in contrast with a traditional algorithm which produces identically the same result for a given starting condition). In the case of evolutionary algorithms, it is customary to undertake multiple runs and report the mean and other descriptive statistics for the outcomes.

Many real-world optimization problems have more than one optimal or extremal point. At an extremum, the first order necessary conditions (FOCs) for a minimum or maximum are satisfied. In the case of a traditional calculus based algorithm, the specific extrema identified by the algorithm depends primarily on the starting conditions specified by the analyst. These types of functions are the bane of researchers everywhere! In the absence of detailed knowledge about the optimal surface, the usual procedure is to restart the traditional algorithm at many different points in the solution space and search for the global optimum point. Problems which exhibit multiple local optima can often be solved by these calculus based methods. However, there is no theoretical or practical way to

guarantee the solution identified by the researcher is the global solution to the problem.

Evolutionary algorithms are sometimes described as global optimizers owing to their well-documented ability to identify the global optima within the given search space. Notwithstanding the published glowing reports, an equal body of published evidence suggests this behavior is not universally observed. Furthermore, it cannot be proved theoretically that they can be relied upon to identify the global best solution. It is most certainly true that relative to traditional algorithms, evolutionary programs carry more solutions through the iteration process and have much greater exploratory ability. These two characteristics enable evolutionary algorithms to more exhaustively traverse the solution space. Consequently, they are much more likely than traditional algorithms to identify the global optima.

Traditional optimization algorithms make heavy use of vectors, matrices and linear algebra operations, which themselves exact a huge computer memory overhead. Consequently, traditional optimization algorithms require extensive amounts of computer memory, especially for the solution of sizable problems. As little as ten years ago the practical usage of traditional optimization algorithms was restricted by the amount of physical and virtual memory addressable by existing microcomputers. In contrast, evolutionary algorithms do not make use of vectors, matrices or other advanced mathematical structures or operators. Their memory requirements are quite modest for similar size problems.

In cases where they can be applied, traditional calculus based optimization algorithms are known for their rapid converge properties. This is especially true in the case of convex functions with linear constraints. Experiments show that for traditional optimization algorithms, the initial phases of search are quite slow. Once they have identified the region where the optima resides, local convergence to the final solution is often very fast. Evolutionary algorithms on the other hand, exhibit behavior which is very much the opposite. Experiments on evolutionary algorithms demonstrate the initial search phase is very fast—the algorithms quickly and efficiently locate the region of the optima. However, the local convergence of these algorithms is slow, in some cases, painfully so. Typically, large amounts of time are required for the population to converge on an optimal point, after the region where it is located has been isolated.

The computational resources required by traditional calculus based algorithms and evolutionary algorithms differ profoundly. Not surprisingly, the time required to achieve convergence is vastly different. Traditional algorithms require large amounts of memory but typically require less than 100 major iterations to converge to a solution. Evolutionary algorithms often require thousands or tens of thousands of iterations to converge to a solution. While it is true that evolutionary algorithms utilize only primitive mathematical operations—it is no understatement to say they do so intensively! Prior to the advent of

microcomputers, the lack of sufficient computing power and sheer cost of computer resources precluded the use of evolutionary algorithms for civilian purposes.

One of the advantages of evolutionary algorithms is their ease of implementation. Unlike traditional algorithms, effective cutting-edge evolutionary algorithms are routinely developed by researchers and hobbyists. As of December 2010, a number of toolboxes and working computer codes are available. Even so, many researchers with limited resources, develop research grade evolutionary algorithms using high level computer languages such as C++, C, Fortran, Java, Visual Basic and Delphi. This is rarely the case for traditional calculus based algorithms.

Heuristics and Microcomputers

Heuristic algorithms are computationally intensive and scientific advances in heuristics are necessarily related to the nearly unimaginable innovations in computer technology made within the last thirty years. Arguably, there are two fundamental aspects of this evolution—vast improvements in computational speed, and the widespread availability of microcomputers.

Although this fact is often overlooked by the young, computers are a relatively recent invention. Depending on the source, the first fully programmable computer was debuted in the 1940s. These early computers were large centralized hardware installations which we now describe as "mainframe" computer architectures. Relative to the current norms, they were incredibly expensive, slow and ponderous. Access to the then existing computational resources was rationed and limited to a few elite civilian researchers, and members of the defense establishment. Experiments using unproven technologies or computationally intensive processes were exceedingly rare.

The advent of microcomputers changed this paradigm. The Apple II personal computer was introduced in 1977 and the International Business Machines (IBM) company marketed their first computer in 1981. Microcomputers were designed to be used independently of institutional controls and shared usage constraints. They could be purchased relatively cheaply by individual researchers, and perhaps most importantly-- were consistently and conveniently available for use. Even the early microcomputers were technically and numerically capable tools. As further technological innovations were made, hardware costs (memory and storage) fell dramatically and computational speeds increased. These characteristics made it possible for established mainstream researchers, as well as hobbyists and researchers working at the fringes of established theory and practice, to purchase microcomputers and to experiment with their ideas freely and at little cost.

Modern researchers often take for granted the massive computational power at their disposal. Since this is particularly true in the case of younger researchers, a brief divergence will provide a useful point of reference. The pace of hardware and speed improvements since the appearance of personal computers has been dizzying. For example, the Apollo 11 mission in 1969, which successfully landed men on the moon, used an onboard computer which had eight times less memory and ran at a much lower speed than the IBM XT personal computer released in 1981 (Robertson 2009). The basic configuration of a 1981 IBM XT computer had 16 kilobytes (0.000016 gigabytes) of random access memory (RAM) and no hard disk. The 1983 IBM XT computer added 10 megabytes (0.010 gigabytes) of hard drive storage and also ran at a central processor unit (CPU) clock speed of 4.077 megahertz (0.004077 gigahertz) (for detailed historical reference, see http://ptgmedia.pearsoncmg.com/imprint_downloads/informit/que/urpc18/OriginalPCReference.pdf). By way of a modern comparison, the laptop used for writing this document operates at a CPU speed of 2.66 gigahertz, a 652.4 fold clock speed increase relative to the 1981 IBM XT. This medium-price range laptop also has an addressable memory space of 4 gigabytes, over 250,000 times larger than the 1981 IBM XT, and hard disk storage of 150 gigabytes, which is 15,000 times more disk storage space.

The birth of heuristic optimization algorithms is inextricably tied to the rise of the microcomputer. Most certainly, the spread of microcomputers and their computational capability provided the essential tools for heuristic algorithm development. Conceptual approaches which had here-to-for been theoretical constructions, could be coded and tested. And they were. Not surprisingly, the description of many heuristic optimization algorithms dates back to this time. Examples include the development of genetic algorithms (1977), the description of particle swarm optimization (1995), simulated annealing (1983) and differential evolution (1995).

The cost of computer hardware, computer software and computer time no longer place an upper limit on the scale or scope of research agendas. The relaxing of these constraints has unleashed many different threads of research on heuristic optimization algorithms. Attitudes about computer resources used in research have also changed. Computational cost is now primarily a question of researcher patience. It is of little consequence to many researchers if their personal computer runs ten seconds, ten minutes or ten hours to achieve a solution. Improvements in the available computational tools, their low cost and near-universal availability have given rise to golden age of heuristic optimization research!

EAs in the Wild—An Update

Evolutionary algorithms (EAs) belong to a larger class of algorithms best described as being inspired by natural phenomenon, particularly the behavior of different organisms. These are often called nature based, nature inspired, or in

some cases, biological algorithms. The universe of nature inspired algorithms is large and creative. Nature inspired algorithms span the realm from bacteria (Kim, Abraham and Cho 2007), to fireflies (Yang 2009), water drops (Shah-Hosseini 2009), ants (Dorigo and Stutzle 2004), beyond the horizon scanning (Chand and Sugianto 2008) and beyond. Newly described algorithms appear in the literature on a regular basis. A notable new entry is the Imperialist Competition Algorithm, described by an Iranian research group (Rabiei, Soroudi and Mohammadi 2011). A selection of the more common and better documented nature inspired algorithms is shown in Table 4.

Table 4.—Selected Nature-Inspired Optimization Algorithms

Algorithm	References
Ant colony optimization (ACO)	Dorigo and Stutzle (2004)
Artificial immune system optimization	Cutello and Nicosia (2002)
Bacterial foraging optimization	Kim, Abraham and Cho (2007)
Bee optimization	Karaboga and Bosturk (2007); Pham et al (2006)
Cuckoo algorithm	Yang and Deb (2009, 2010)
Differential evolution (DE)	Storn and Price (1995, 1997)
Firefly optimization	Yang (2010)
Fish optimization	Huang and Zhou (2008)
Genetic algorithms (GA)	Haupt and Haupt (2004)
Horizon scan	Chand and Sugianto (2008)
Particle swarm optimization (PSO)	Eberhart and Kennedy (1995); Kennedy and Eberhart (2001)
Water drop optimization	Shah-Hosseini (2009)
Simulated annealing	Kirkpatrick, Gelatt, and Vecchi (1983)

The evolutionary algorithms, including genetic algorithms, particle swarm optimization, artificial bee colony optimization and differential evolution are a sub-category of the nature inspired optimization algorithms. Evolutionary algorithms and their characteristics are the focus of this research and are discussed in greater detail in subsequent sections of this document.

Research on nature inspired algorithms is ongoing and active. There have been several evaluations and performance comparisons of nature inspired algorithms. These have typically focused on the less-esoteric members of this algorithm class. The most expansive of these evaluations is found in the book by Wahde (2008). Readily obtainable studies by Potter et al (2009) and Mezura-Montes and Lopez-Ramirez (2007) are also very useful contributions to this line of research.

Related Algorithms

Hybrid Algorithms

Hybrid evolutionary algorithms are frequently and routinely encountered in the applied literature. As distinct from memetic algorithms, which combine evolutionary algorithms and traditional (calculus based) algorithms, hybrid algorithms are constructed from two or more evolutionary algorithms. The resultant hybrid algorithm is often described as being (potentially) superior to either parent, especially in certain specific problem domains.

Hybrid combinations of nearly every evolutionary algorithm have been described. There are a plethora of hybrid combinations for PSO, DE, ACO and GA's and there are also hybrid combinations of other less well-known algorithms such as bee algorithms. Engelbrecht (2005) reviews several hybrid PSO algorithms including GA based PSO, DE based PSO (also see Liu, Cai and Wang (2008)) and ACO based PSO. Banks, Vincent and Anyakoha (2008) review about 25 different hybrids and Neri and Tirronen (2010) cite about 30 more. A quick electronic perusal of the recent literature reveals a remarkable number of hybrid combinations and variants thereof.

At least some part of this activity may be driven by the need for researchers to differentiate their publication products. Even so, based on the existing number of different hybrids, this appears to be an incredibly fertile topical area for future research.

Memetic Algorithms

Memetic algorithms harness the global search characteristics of evolutionary algorithms with the fast local search properties of traditional (calculus based) optimization methods. Evolutionary algorithms such as PSO, DE and RCGA are able to rapidly and efficiently locate the neighborhood of the global optima, or a set of candidate optima. Typically however, their local convergence properties are rather slow. Evolutionary algorithms spend a disproportionate amount of time achieving convergence, after the neighborhood of the optima has been identified. Memetic algorithms utilize evolutionary algorithms to identify the neighborhood of an optimal point and then pass control of the optimization process to a traditional algorithm.

Engelbrecht (2005) reviews several PSO based memetic algorithms including hill-climbing PSO, stochastic local search PSO and gradient based PSO approaches. Additionally, there are a number of relatively comprehensive studies of memetic approaches. Particularly revealing are studies of GA based memetic algorithms (Li, Ong, Le and Gob 2008), DE based memetic algorithms (Neri and Tirronen 2010) and a comparison of different evolutionary based approaches (Nguyen, Ong

and Krasnogor 2007). Based on the available evidence, this two-step approach is quite efficient for continuous, differentiable functions and has the potential to stimulate many related threads of research.

EA Selection Process

Algorithm Selection

The selection of specific algorithms for this research was informed by the existing professional published and grey literature, applications to similar problems, performance reports and several practical considerations. As described in Table 4, the universe of evolutionary algorithms described in the literature is diverse and growing at a very rapid pace. Potentially, a number of different evolutionary algorithms could be applied to hydropower dynamic economic dispatch and unit dispatch problems. As with any research effort, this one was constrained by the resources available; primarily funding and researcher time. These and other practical constraints dictated, to some extent, the range and number of algorithms which could be explored.

Candidate Selection

An initial preliminary literature review was undertaken to identify candidate evolutionary algorithms for use in this research. The initial literature exploration was followed by a relatively extensive review of the power engineering literature with a focus on identifying intersections between the candidate algorithms and previous applications to electric power system problems. Subsequently, a more intensive review of the recent literature pertinent to specific candidate algorithms was conducted.

The literature review process resulted in identification of five candidate algorithms. These algorithms were; particle swarm optimization (PSO), genetic algorithms (GA), differential evolution (DE), ant colony optimization (ACO) and the bees algorithm (BA).

Selection Criteria

Selection of evolutionary algorithms for this research project required some artistry and judgment. One factor which weighed heavily in the selection process was the depth and breadth of previous applications. The widespread use of a particular algorithm and the number of examples where it has been applied to a particular class of optimization problem provides some evidence of the algorithm's efficacy and potential for application in other arenas. For example,

both GA and PSO have been very extensively applied to an amazing variety of problem types. In contrast, the literature on the dragon-fly algorithm consists of a small handful of specific applications, the bulk of which are by the same author. With limited investigational resources to devote, the decision to eliminate the latter from consideration was rather straightforward

The hydropower dynamic economic dispatch problem and the unit dispatch problem are both examples of constrained optimization problems. For this reason, a substantial part of the decision process was focused on evolutionary algorithms which had been applied to the general class of constrained optimization problems. Although many of the evolutionary algorithms listed in Table 4 could potentially be modified, in some way, to accommodate constraints, it seemed prudent to limit the search to algorithms for which published examples existed. This eliminated some relatively promising algorithms from the subset of algorithms retained for further investigation. The artificial bee colony optimization (ABCO) algorithm, for example, is a new and seemingly quite efficient evolutionary algorithm. During Phase 1 of this research effort, there were no known examples of the ABCO algorithm applied to constrained optimization problems. Consequently, the ABCO algorithm was not selected for study in Phase 1. Subsequently however, several applications of the ABCO algorithm to constrained optimization problems have emerged and appear quite promising. As a result, this algorithm was considered for investigation in this Phase 2 research effort.

Finally, the choice of evolutionary algorithms was further limited to those algorithms designed for the continuous real number domain. Although many applied problems can be specified in discrete forms (in fact, all continuous problems can be re-specified as discrete approximations), the most natural and appealing choice for solving a continuous real-valued problem is to use an algorithm which operates in the continuous real-valued domain. Ant colony optimization (ACO) is certainly a promising candidate algorithm, but is primarily useful in the discrete domain. For this reason, ACO was eliminated from further consideration.

Algorithms Selected

Based on the multiphase literature review, previous application to constrained optimization problems and limiting the choices to continuous real-valued algorithms resulted in a relatively small subset of evolutionary algorithms which were retained for detailed investigation. This subset includes; RCGA, DE, PSO and ABCO. A short description of each of these algorithms follows while the details of these four algorithms are described more fully in the Phase 1 Report (Harpman 2012, Appendices 7, 8, 9 and 10) and in Appendix 1 of this report.

Real Coded Genetic Algorithm (RCGA)

Genetic algorithms were the first of the evolutionary algorithms to be described in the literature. They use techniques inspired by evolutionary biology including inheritance, mutation, selection, and crossover. This research focuses on the less-studied real coded genetic algorithm which is faster and more naturally applied to the dynamic economic dispatch and the unit dispatch problem, than the binary variant.

Genetic algorithms are based on virtual populations which are termed individuals (or phenotypes). For each generation or iteration, the fitness of every individual in the population is evaluated and the most fit individuals are selected and modified (recombined and possibly randomly mutated) to form a new population. The new population survives into the next generation or iteration of the algorithm. The algorithm terminates when a satisfactory fitness level has been achieved or the maximum number of iterations has occurred. Appendix 8 in Harpman (2012) contains a complete description of RCGA.

Differential Evolution (DE)

Differential evolution (DE) was jointly developed by Storn and Price (1995, 1997) and is one of the more recently described global heuristic optimization methods. In many respects, it resembles a simplified form of genetic algorithm, albeit with several distinct and highly desirable performance characteristics.

The DE approach is based on a virtual population of np-independent individuals. During each generation, these individuals reproduce and undergo selection. Only the fittest individuals in the population survive to reproduce in the next generation. Over successive generations, the population becomes increasingly fit —thereby identifying the optimum (minimum or maximum) of a function. DE is described in considerably more detail in Harpman (2012, Appendix 9).

Particle Swarm Optimization (PSO)

PSO is a global heuristic optimization method. It was invented by Kennedy and Eberhart (1995) who developed the concept by observing the behavior of flocking birds. PSO is classified as a stochastic, population-based evolutionary computer algorithm for problem solving.

PSO utilizes np-independent virtual particles, which "fly" through the search domain, have a memory and are able to communicate with other members of their "swarm." Each particle has only one purpose—to identify the optimum (minimum or maximum) of a function within the feasible search space. PSO is described in more detail in Appendix 10 of the Phase 1 Report (Harpman 2012).

Artificial Bee Colony Optimization (ABCO)

The artificial bee colony optimization (ABCO) algorithm is among the most recently (circa 2001) described algorithms to appear in the literature. ABCO is an optimization approach which is based on the collective behaviour of a hive of honey bees, searching for food. ABCO utilizes employed bees and unemployed bees, the latter composed of onlooker bees and scouts, to locate the best food sources in the search space, thereby identifying the optima of a function.

Although there are a number of bee related algorithms, many of these are for discrete problem types. Karaboga et al (2012) reported a useful application of what he terms the artificial bee colony optimization (ABCO) algorithm to constrained optimization problems. He and his disciples followed with a series of applied applications. A review and assessment by Mezura-Montes and Cetina Dominguez (2012), suggests the artificial bee colony optimization (ABCO) algorithm is the most promising bee algorithm for this research application. The ABCO algorithm and its application to the unit dispatch problem are described in further detail in Appendix 1.

Initialization

The first step in all of the heuristic optimization algorithms is to identify the starting positions of a specified number (np) of particles or individuals in the d-dimensional search space. This process is termed, "initialization."

Purpose and Implications

A large number of empirical studies conclude the choice of initialization strategy and its properties can profoundly influence the outcome of a heuristic optimizer. The successful identification of the global optima is dependent on the proximity of the initial points. To the extent an initialization approach does not adequately cover a particular region in the search space, and this region contains the global optima, the algorithm may fail to identify the global optima. Or, if the chosen initialization method places a number of particles in the region of the search space hosting a local optima, the algorithm may become trapped and converge on the local, rather than the global optima. Second, the number of iterations, the computational effort required and the convergence time required are related to the proximity of the initialized points to the optima. Finally, to the extent the initialization process is stochastic, the point of algorithm convergence, local versus global extrema and the precision of convergence will also vary.

Random

The vast majority of applied work employs the uniform random distribution to initially locate points in the search space. The upper left plot shown in Figure 7 illustrates a random initialization of np=250 points in 2-dimensional bounded [1, 1] space.

Inspection of the upper left-hand plot in Figure 7 suggest the points are considerably more numerous in some regions of the search area than in other regions. This is typical of random initialization methods, which often result in a non-systematic location of the initialized points within the bounded area. Clerc (2008) concisely describes this phenomenon in the first section title of a widely cited paper as, "Uniform Random Distributions: Easy, but Bad."

One frequent contributor to poor random initialization performance is failure to employ a high quality random number generator (RNG). Random sequences produced by an RNG are an important component of this research. As described in the Phase 1 Report (Harpman 2012, pages 24-27 and Appendices 12 and 13), considerable effort was devoted to identifying and implementing an appropriate RNG for use in this research project.

After carefully considering the available options, a well-proven, if not state-of-the-art RNG was adopted for use in this research project. All of the algorithms developed during this research effort employ a Delphi coded implementation of the Mersenne Twister RNG (Matsumoto and Nishimura 1998) developed by David Butler and obtained from the SourceForge Library, http://fundementals.sourceforge.net/units.html. This algorithm is also known by its Association for Computational Machinery (ACM) identification number as algorithm MT19937.

Sequences

A number of researchers have proposed the use of low discrepancy sequences for initialization purposes. Low discrepancy sequences are also called quasi-random or sub-random sequences. The points in these sequences are said to be more systematically located in the search space, with fewer gaps and more equal spacing between points. Low discrepancy sequences in the EA literature include the Sobol (Pant, Thangaraj, Singh and Abraham 2008), Van der Corput (Pant, Thangaraj and Abraham 2009) and Halton (Uy, Hoai, McKay and Tuan 2007) sequences, to name but a few.

This research thread is based on the mathematical properties of low discrepancy sequences which allow them to more exhaustively and systematically span the d-dimensional search space. While these properties can be statistically demonstrated, the visual approach provides much the same intuition. Figure 7

shows plots of the first 250 points in 2-dimensions over the range (0,1) generated using the Mersenne Twister RNG (described further in Harpman (2012 Appendix 12), the Neiderreiter sequence, the Weyl sequence and the Haber sequence.

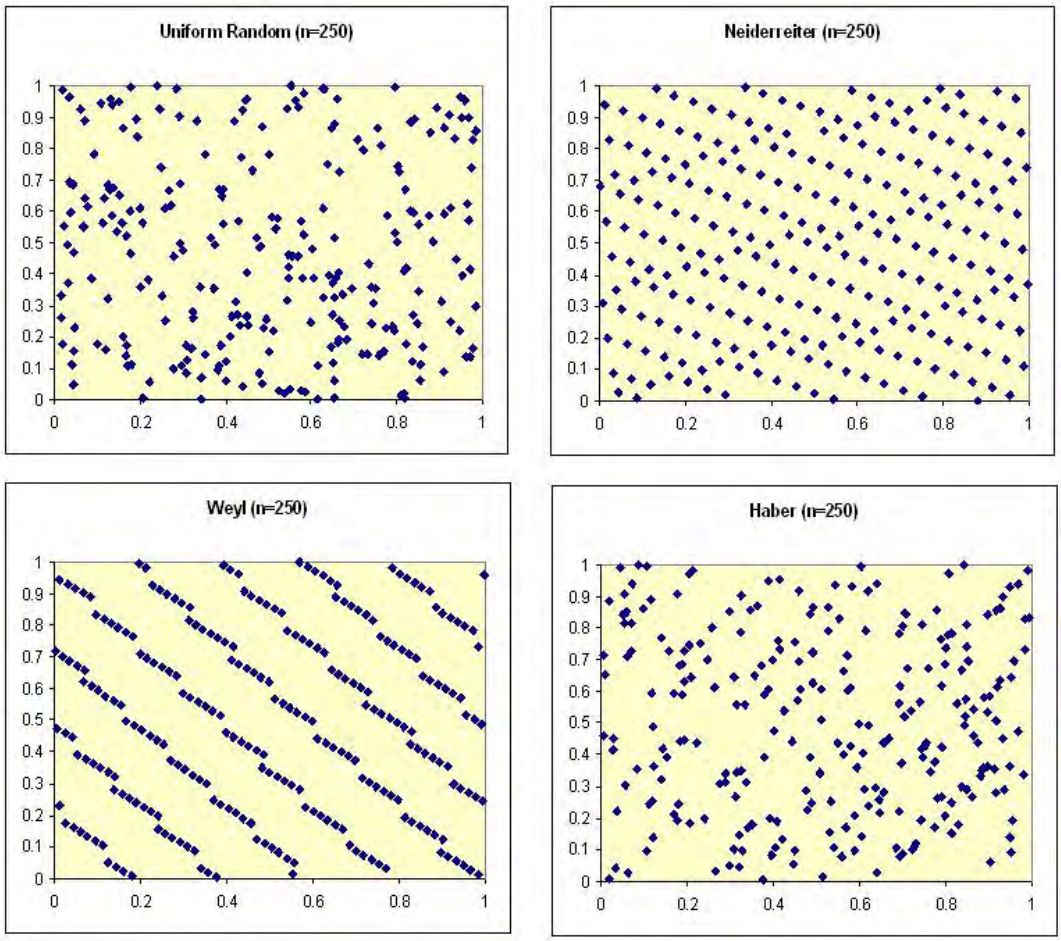

Figure 7.—First 250 Points Generated by Four RNG Methods.

One component of this research effort was to investigate the potentials advantages of employing some of these low discrepancy sequences. As part of that effort, computer codes for the Neiderreiter, Weyl, Haber, Halton and Torus sequences were developed. This code was based on MatLab code from the EconToolbox which accompanies Miranda and Fackler (2006). The prime numbers utilized in coding these sequences were drawn from Caldwell (2009).

Other Methods

Several recent efforts have utilized other logical and mathematical approaches for initialization. A selection of these approaches include the simplex method

(Parsopoulos and Vrahatis 2002), quadratic interpolation (Pant, Singh and Abraham 2009), tessellations (Richards and Ventura 2004) and the opposition method (Rahnamayan, Tizhoosh and Salama 2008, Omran 2009).

The opposition method was originally described by Rahnamayan, Tizhoosh and Salama (2008) and is employed for initialization purposes by Omran (2009), who uses the term opposition based learning (OBL). The OBL procedure improves the starting fitness of the initialized points.

The OBL procedure results in a total population of 2*np individuals, or particles. The fitness of these 2*np particles are assessed and the particles are sorted by their fitness. The np most fit particles are retained and their positions are used to initialize or start the heuristic optimizer. Like the other approaches described here, the OBL method entails some additional implementation complexity and computational burden.

Previous research has focused on the application of different initialization methods to a suite of test problems. Their potential efficacy when applied to the solution of the hydropower problems examined here is unknown. Subsequent sections of this document will report the results of experiments which explore the use of several of these initialization techniques.

Constraints and Constraint Handling

Types of Constraints

Constraints separate the solution space into feasible and infeasible spaces. The resulting feasible solution space is generally limited and can be discontinuous, even when the optimization problem is not. Constraint equations can be linear or nonlinear in nature. In general, there are three classes of constraint equations. These broad classes are; boundary constraints, equality constraints and inequality constraints.

Boundary constraints serve to define the borders of the solution space. Boundary constraints commonly take the form of simple upper or lower bounds on the independent variables. These are called "box" constraints. However, simple bounds are not the only types of boundary constraints. For example, the circumference of a hypersphere (a sphere in multi-dimensional space) is also a boundary constraint. An example of a simple lower bound constraint is shown in equation (1).

(1) $$x_i > c$$

Equality constraints specify that a function of the independent variables is equal to a scalar constant. For example the amount of electricity generated (supplied) at a given instant must be equal to the amount of electricity demanded at that time. An example of an equality constraint is shown in equation (2).

$$x_1 + x_2 + x_3 = k \tag{2}$$

Inequality constraints specify that a function of the independent variables must be greater than or equal to, or less than or equal to, a given scalar constant. For example, the contents of a reservoir must be less than or equal to the storage capacity of the reservoir. A simple example of an inequality constraint is shown in equation (3).

$$v_t \leq f(x) \tag{3}$$

Constraint Handling

Research on the incorporation of constraints in evolutionary programming methods and the solution of constrained optimization problems with evolutionary methods is rather voluminous. Carlos Coello Coello published a widely cited synopsis of this work (Coello Coello 2002) and maintains an online annotated bibliography summarizing this ever-expanding body of research (http://www.cs.cinvestav.mx/~constraint/index.html). As of February 2012, this bibliography exceeded 115 pages in length.

Generalizing rather broadly from the literature, constraint handling techniques can be classified into six categories. These are problem reformulation, rejection of infeasible solutions, penalty approaches, feasibility preserving methods, repair methods and mixed approaches. The Phase 1 Report (Harpman 2012) reviewed these constraint handling technologies in some detail and this discussion is not repeated here.

This research application, in common with most applications of evolutionary algorithms, employs a combination of all of the constraint handling methods described in Harpman (2012). This is referred to as a "mixed" constraint handling approach. The algorithms tested here employ penalty functions, repair methods and reject some types of infeasible solutions. As described in Appendix 8, a novel triangular penalty approach was developed to avoid operations inside of prohibited operating or rough zones.

Fitness Comparisons with Constraints

Pair wise comparison of two solutions to identify which is the most fit is relatively straightforward for unconstrained optimization problems. In the case of constrained optimization problems, such comparisons are made considerably

more complex because of the potentially confounding influence of the penalty function. To illustrate this problem more fully, first recall that when applied to a maximization problem, fitness (F) is often defined as the sum of the objective function value (f(x) *minus* the infeasibility penalty (P), if any.

(4) $$F=f(x) - P.$$

It may also be useful for the discussion which follows to recall that for constrained maximization problems, an infeasible solution is typically one in which one or more variables exceed their upper bound restrictions. By definition, the objective function value in these cases is higher than it would be if the solution were feasible.

Typically, the analyst may spend considerable time understanding the nuances of their particular optimization problem and judiciously selecting the values of the penalty function parameters. This systematic approach will help the analyst to properly scale the penalty function value in relation to the objective function values. Even so, identification of the "most fit" solution in a pair wise comparison remains problematic. The operative question in such cases being--does the (negative) value of the penalty function outweigh the objective function value? What if the penalty is rather small compared to the value of the objective function? In recognition of this logical and mathematical dilemma, most applications of evolutionary algorithms utilize an oft-cited work on this subject by Deb (2000).

Following Deb (2000), for any pair wise comparison of solutions, there are three possible cases. These are; (1) both solutions are feasible (the penalty is zero), (2) one solution is feasible (the penalty is zero) and the other solution is not feasible (the penalty is nonzero), and, (3) both solutions are not feasible (the penalties are both nonzero). Deb (2000) devised a comparison scheme for selecting the most-fit solution under each of these cases. This scheme is described in the bulleted list which follows.

- If both solutions are feasible, select the solution with the greatest fitness. Owing to the fact the penalty is zero for both solutions; this is equivalent to selecting the solution with the highest value of the objective function.

- If one solution is feasible and the other solution is not feasible, then select the feasible solution without regard to its fitness value. It is useful to note that under this criteria, the value of the objective function is not a factor in the decision process.

- If both solutions are infeasible, select the solution which has the lowest value of the penalty function (the most feasible solution). Once again, the value of the objective function does not play a role in this decision.

The selection scheme devised by Deb is straightforward and readily implemented in code. It has found wide-spread application and acceptance in evolutionary

algorithms. Although it is not entirely foolproof, it is often used and (very) often cited.

Performance Measures

Algorithm Performance Metrics

Ascertaining the success of an evolutionary algorithm in identifying the solution to an optimization problem can be a rather subjective undertaking. Furthermore, discerning real, rather than apparent, differences between two evolutionary algorithms can be especially problematic. These difficulties arise for two disparate reasons: (1) the characteristics of real-life optimization problems, and, (2) the nature of evolutionary algorithms. Some further explanation will help to put both of these subjects in perspective.

Practical Optimization Problems

Most readers of this document are familiar with solving textbook example optimization problems. The majority of these problems are convex, and each has a single known optimal solution. Generally, the objective is to find the optima of such problems, typically with a traditional, calculus based approach. Identifying the minimum or maximum point is often undertaken analytically, for relatively simple textbook problems. More complex problems are attacked with a variety of numeric methods, such as the Newton Raphson described in Harpman (2012 Appendix 6). For numeric methods, the goal is to efficiently and reliably identify the optimal point to within some acceptable level of numeric precision.

In contrast to textbook optimization problems, many practical optimization problems have unknown optimal points. [If their solutions were known, there would be no need for algorithms to solve them]. To state the obvious point, there is no way to know when the optima has been found. Complex, ill-behaved problems with multiple local optima are relatively common in applied efforts. Algorithms may converge on a particular local optima, or may converge on a different local optima when started from varying initial positions. This gives rise to a further complexity—identifying which, if any, of the identified local optima is the maxima or minima of the function.

Nature of Evolutionary Algorithms

The inherent nature of evolutionary algorithms can obscure the attainment of the optima and certainly makes it much more difficult to discern between two competing candidate algorithms. First, unlike calculus based solution approaches,

evolutionary algorithms carry multiple solutions throughout the iteration process. For example, EA's carry one solution with each member of the population. In the population, some of these solutions are inferior solutions and one of them is the "best" solution. Furthermore, these solutions vary with each application of the algorithm. For any given algorithm trial, the np solutions are randomly initialized within the search space. By random chance, some of the initialized solutions may land in the feasible solution region, or perhaps not. The specific initialization process and the initialization itself give rise to varying degrees of progress towards a solution. Likewise the stochastic nature of the solution algorithm, as manifested at each generation or iteration, has an influence on the algorithms rate of progress towards identification of the optima. For one trial, a series of fortuitously generated random values may result in a rapid convergence on the optima. For a different trial, a series of unfortunately generated random values may result in a failure to converge, a premature convergence, spurious convergence or a lengthy convergence to the optima. Consequently, evolutionary algorithms may return different solutions for multiple independent trials, even when applied to the same problem. Clearly, the convergence behavior of an evolutionary algorithm will vary with each trial or experiment.

Multiple Trial Approach

Owing to the complexities of most practical optimization problems and the inherent characteristics of evolutionary algorithms, a multiple or replicated trial approach is typically employed to gauge their success and compare efficacy between two candidate algorithms. A trial is one independent application of the algorithm to a specific problem. Typically a pre-set number of trials, for example 50 replications, are carried out on the same problem and selected measures of success are extracted for each of the trials. At each trial, a new initialization of the population occurs and a new random sequence is generated. In aggregate, the resultant measures of success then serve as a more appropriate and informative gauge of algorithm success. Formal statistical analysis of replicated success measures, compared across candidate algorithms, allows for reasoned selection of more effective algorithms.

Common Measures of Performance

Although there are many possible performance metrics, four measures are most commonly encountered in the literature. These are accuracy, reliability, robustness and efficiency. Other metrics including diversity and coherence are discussed in texts but seldom encountered in the professional literature.

Accuracy

As might be expected, solution accuracy is of paramount importance in assessing algorithm performance. In the case of functions with a known global best

solution, accuracy is assessed as the difference between the best solution achieved by the algorithm and the known global best solution, at a particular number of iterations. For functions with unknown global optima, accuracy is the fitness of the global best solution attained by the algorithm over a given number of iterations.

If the accuracy of two different evolutionary algorithms is being compared, the usual practice is to compare this metric for the same number of function evaluations (FE's) rather than iterations. This is said to provide a better basis for comparison since some algorithms may require more per-iteration function evaluations than others, thus being more computationally intensive and, in the process, obtaining more information about the search space.

When the derivative of the function can be computed, derivative information can be used to assess the quality of the solution achieved. If the derivative can be computed, it ought to be zero, or very near to zero, at the optimal point identified by the algorithm. Of course, the derivative will be zero at any stationary point, including both global and local optima. For this reason derivative information is not entirely informative.

Reliability

Algorithm reliability is of great importance both to researchers and practitioners. The greater the certainty that an algorithm will (a) converge, and, (b) converge on the global optima—the more the more useful the algorithm is. For evolutionary algorithms, reliability may be assessed by measuring the percent of solutions which fall within an acceptable tolerance of the known global optima, for a given number of iterations. Or, when prior knowledge of the function is unavailable, as the percent of solutions which converge to a specified tolerance for some specified number of iterations. This metric is especially applicable to highly complex functions for which convergence is less common and somewhat less informative for better behaved functions for which convergence is routine.

Robustness

Robustness is a term used to describe the variance around a particular performance criteria. The variance around a success metric is a measure of dispersion. The smaller the variance over some given number of iterations, the more robust or stable the algorithm is judged to be.

Efficiency

Efficiency is a measure of the resource cost or effort incurred to achieve a solution with a desired level of accuracy. Efficiency is typically measured in terms of the number of iterations (or generations) required by the algorithm, the central processor time (CPU) time required, or the number of function evaluations (FE's) required to achieve a solution.

In the context of evolutionary algorithms, efficiency is a particularly relevant performance metric and is especially telling relative to traditional optimization approaches. Evolutionary algorithms are known for being computationally intensive and requiring relatively long computational efforts to achieve solutions. For functions which can be solved with traditional calculus based approaches, traditional solution approaches can usually find the solution much faster than evolutionary algorithms.

Algorithm Stopping Criteria

Introduction

The preponderance of numerical optimization algorithms are based on some sort of iterative or repetitive procedure. An important aspect of these algorithms is the design of intelligent convergence or stopping rules. These rules detect when the routines have converged on a solution, and then halt the iterative process.

The Trade-Off

The design of stopping rules necessarily requires an explicit trade-off between computational cost (a function of the number of iterations and hence, time) and solution accuracy. At best, numerical optimization algorithms can provide an approximation of the true solution vector, not the exact solution. In general, the numerical accuracy of the solution vector is improved with each succeeding iteration. Theoretically, a numerical algorithm can identify the exact solution in an infinite number of iterations. In more technical terms, these algorithms can be shown to achieve the true solution only asymptotically. Luckily, most research requirements can be satisfied by an answer that is "close enough" to the true solution and is available in a finite timeframe. Two interlinked questions emerge. How close is "close enough," and, what is an acceptable computational cost?

In many optimization applications, the scale and nature of the problem will suggest an appropriate level of accuracy. In many financial applications, for example, an absolute accuracy of $1.00 (the nearest dollar) or $0.01 (the nearest cent) is more than sufficient. In other cases, accuracy requirements are less clear cut. Almost all numerical methods texts include a discussion of this subject. Interestingly enough, the specifics of computer hardware and software design limit the number of significant digits of accuracy which can be achieved. This places an upper bound on the how close is "close enough" question. Press, et al (1999) and Judge (1998), among others, have useful discussions of these limitations on numeric accuracy. Press, et al (1999) has an especially useful discussion of this topic and the roles that data type, word length and register size play. As a rule of thumb, both Press, et al (1999) and Judge (1998) admonish the

researcher not to specify an accuracy level greater than the square root of the machine accuracy. For most computers with a 32-bit word length, machine accuracy is around 3×10^{-8}. This suggests that a tolerance level (δ, ε) of around 1.7320×10^{-4} is about the best that can reasonably be expected.

In the early days of computer assisted research, computational cost was a much more important consideration than it is today. At the dawn of the computer age, research teams were quite literally charged for each millisecond of computer time they used. Because computer hardware was both expensive and rare, researchers paid for, or were allocated, a computer budget. Other researchers depended on the same hardware and research researchers dared not exceed their computer budgets, or severe sanctions were levied.

In modern times, the widespread availability of microcomputers, their speed and their relatively low cost, combine to make computational cost a less-important consideration. Computational cost is now primarily a question of researcher patience, rather than a funding issue. To most researchers, it is unimportant if the computer runs ten seconds, ten minutes or ten hours to reach a solution (as long as it does so). If long run-times are anticipated, it may prove convenient to schedule an overnight computer run. Some routine mathematical simulations are expected to take several hours, to a day or more to complete. A decade ago, computational costs of this magnitude were an unimaginable research luxury!

Calculus Based Criteria

Typically, convergence criteria for calculus based optimization algorithms are based on the first order conditions for an extrema—which require the first derivative to be equal to zero. In the multivariate optimization context, the first order conditions require the gradient vector to equal zero. As a practical matter, the norm of the gradient vector is evaluated to detect when this has occurred.

Judd (1999, p. 104) provides a concise and straightforward explanation of a two-fold stopping criteria or rule. First, a test is applied to identify whether or not the solution vector is changing significantly between iterations. Second, a test is applied to identify whether or not the first order conditions are met. This combined approach is shown in equations (5) and (6).

(5) $$\left\| x_{n+1} - x_n \right\| < \varepsilon(1 + \left\| x_n \right\|)$$

Where: n = iteration number
x = solution vector
ε = convergence criteria
$\left\| \text{m} \right\|$ = norm of the vector m.

Equation (5) compares the norm of the difference between solution vectors at two different iterations with epsilon (ε) times one plus the norm of the solution vector from the last iteration. This part of the stopping rule identifies whether or not the solution vectors achieved at two different iterations are the same, or approximately so. The formulation guards against division by zero and allows for the researcher to set some desired level of ε for detecting when this has occurred.

If the first part of the convergence test is satisfied, the next step is to see if the solution vector at iteration (n) satisfies the first order conditions for an optimum. As might be expected, this test focuses on whether the gradient vector is zero, or approximately so. This part of the stopping rule is described by equation (6).

$$\left\|\nabla f(x_n)\right\| \le \delta(1+\left|f(x_n)\right|) \tag{6}$$

Where: n = iteration number
x = solution vector
$\nabla f(x)$ = gradient of the function
δ = convergence criteria.
$\|m\|$ = norm of the vector m.
|f(x)| = absolute value of the solution.

Again, this well-devised formulation guards against the possibility f(x)≈0 and a possible division by zero.

If both parts of the converge test (equations 5 and 6) are satisfied, the solution vector has converged to an approximate optimal point. If the solution vector (equation 5) has converged, but the first order conditions are not met (equation 6), the solution has converged, but not near an optima.

Convergence tolerances, epsilon (ε) and gamma (δ) are used to test when this rule is satisfied. These tolerances are set by the analyst. Both of these control parameters are commonly encountered in optimization routines and, as discussed in Judd (1999), Press et al (1998) and elsewhere, are limited by the ability of the computer platform to characterize real numbers.

Criteria for Evolutionary Algorithms

The stopping criteria employed for traditional calculus based optimization procedures are not applicable to evolutionary algorithms. There are two reasons for this. First, evolutionary algorithms are multiple solution methods; they carry a number of solutions throughout the iterative computation process. In the case of particle swarm optimization (PSO), for example, each member of the swarm

stores its (own) personal best solution. If the swarm size is n=40, forty solutions are maintained and iteratively improved during the lifespan of the swarm. In contrast, calculus based optimization algorithms carry only a single solution throughout the computation process.

Secondly, the stopping or convergence criteria for traditional calculus based optimization approaches, not surprisingly, rely on calculus concepts (e.g. derivatives, gradients, hessians, etc). Evolutionary algorithms require only primitive mathematical structures, do not need and generally eschew advanced mathematical constructs, such as derivatives. In fact, their derivative-free nature is often touted as one of the advantages of these algorithms. Furthermore, evolutionary algorithms are often applied in situations where the underlying functions are discontinuous and ill-behaved. In these cases derivatives for the underlying functions either cannot be analytically derived, or simply don't exist. This makes it impossible to apply the stopping rules used in traditional calculus based optimization approaches.

An ideal stopping rule for evolutionary algorithms represents an acceptable trade-off between computational efficiency and the probability of detecting convergence on the true optima. At the same time, such a rule should minimize the likelihood of prematurely halting the iterations before the optimal point is identified to an acceptable level of precision.

The preponderance of published articles found in the evolutionary algorithm literature employ the maximum number of iterations as a stopping rule. Using this approach, the algorithm proceeds until a pre-set maximum number of iterations have been completed-- then it halts. The "best" solution from the population of solutions is identified and then reported. The primary advantage of this approach is it is simple to implement. This stopping rule is frequently used to compare the behavior of alternative parameter settings and algorithm variants. The disadvantage is profound—the preset maximum number of iterations may or may not correspond to the number of iterations required for algorithm convergence. For example, if the maximum number of iterations is set at 1000 and convergence is achieved at 10 iterations, there are 990 unnecessary iterations. Conversely, if convergence does not occur until 5,500 iterations, the results returned for 1000 iterations will not reflect the optimal solution to the problem. Since evolutionary algorithms are stochastic, their rates of convergence vary in a probabilistic manner. As applied to a given problem, one trial may converge in 50 iterations and another trial in 120 iterations. Without prior knowledge of the problem's convergence behavior, there is no known technique for effectively setting the maximum number of iterations. All of these factors argue against the application of this stopping rule—except for comparative purposes.

Figure 8 illustrates the convergence behavior for the 24-hour dynamic economic dispatch problem solved using differential evolution (DE). This plot shows the mean fitness by iteration for 50 trials. This is a maximization problem and the fitness improves as the number of iterations increases. The green error bars illustrate the variance around the mean solution. As shown, large improvements in

fitness are made initially with the majority of the improvements occurring in the first 50 iterations. Fitness improvements thereafter are made only slowly and at extensive computational cost, relative to the accuracy obtained.

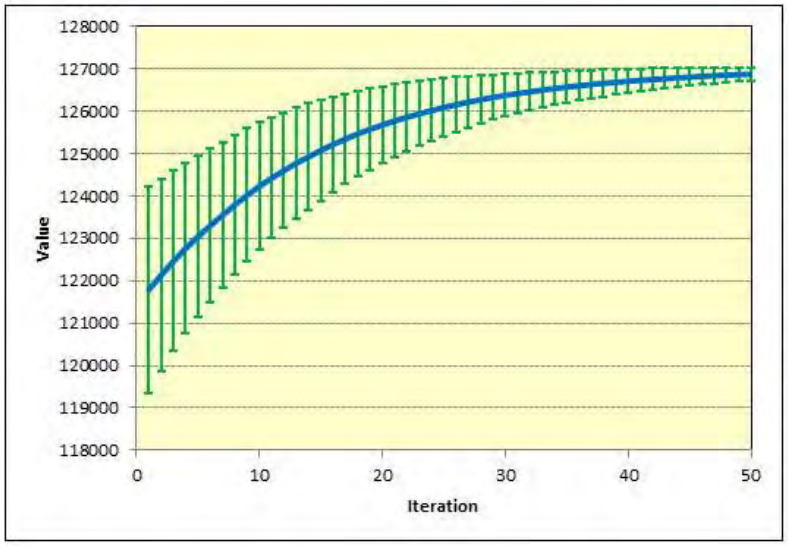

Figure 8.—Convergence Behavior with DE (ntrials=50).

As the number of iterations increases, the estimated solution asymptotically approaches the true solution. For this particular 50 trial experiment, the mean solution at 1,000 iterations is $127,079.25 and at 5,000 iterations, it is $127,097.14. This represents an improvement of about $17.89 (0.14%) at a cost of 4,000 additional iterations, which represents a 400% increase in computational cost.

The subject of stopping rules is not well addressed by the available texts on evolutionary programming such as Engelbrecht (2005), Kennedy and Eberhart (2001). However some of the more recent research efforts have focused on this topic, for example Zielinski et al (2006), Zielinski and Laur (2007) and Zielinski and Laur (2008). The potential efficacy of the suggested approaches when applied to the solution of the hydropower problems examined here is unknown. Consequently, a non-trivial amount of effort was devoted to this subject as part of this research effort. Subsequent sections of this document will report the results of experiments which explore the use of several different stopping or convergence approaches.

Parameters, Tuning, and Variants

This section of the document describes the choice of parameter values and selection of the subset of algorithm variants examined in this research effort. The

simple descriptor, "variants" is used to denote these algorithm variants throughout the remainder of this discussion. Considered in aggregate, these two subjects constitute a substantial portion of the literature devoted to evolutionary algorithms. As this is primarily an applied research effort, a less extensive and less systematic approach was employed.

Evolutionary algorithms have a relatively large number of parameters and approach variants. For example, the size of the population (np) is a user controllable parameter in all of the approaches examined here but each of these algorithms has additional parameters, some of which may interact with each other. Similarly, each of the evolutionary algorithms examined here includes user selectable algorithm variations, such as the neighborhood or global optimization strategies in PSO and the wide range of mutation strategies in DE. Selection of the appropriate value for these parameters and as well as choosing the particular logic, strategy and operational variants are specific to the logic of each evolutionary algorithm. Conclusions about the effects of parameter setting are mixed. Some researchers report effective applications of these algorithms are quite sensitive to parameter choice while others have suggested their efficacy is largely insensitive to the specific combination of parameter values chosen. Other researchers have reported that parameter choice is problem specific. On the topic of algorithm variants, the available evidence is also less than clear. The literature abounds with newly described variations for each of these evolutionary algorithms. Seemingly without exception, each of these variants is stated to dominate the other variants described in the previous literature.

Population Size

All evolutionary algorithms are multiple solution processes. The number of solutions, or population size, influences the performance of these algorithms and their successful application. There is an explicit tradeoff between the size of the population, the number of iterations required to achieve convergence and the computational effort. Many authors use the number of objective function evaluations (NFE's) as a measure of computational effort. For the PSO algorithm, for instance, the NFE's required for each generation is given by the size of the population (np) times the number of iterations (iter) or, NFE = np*iter (disregarding initialization). For a problem of any given dimensionality (dim), the larger the population size, the more likely that one or more of the individuals in the population will be initialized to the vicinity of the global optima in the search space. All else being the same, a larger population might then be expected to require a smaller number of iterations to achieve convergence, converge more rapidly and converge on the global (rather than local) optima. The drawback to large population sizes is that each member of the population must be evaluated at each generation. For a complex objective function, with a larger number of dimensions, this can greatly increase the computational effort, requiring significantly longer solution times.

RCGA Parameters

The basic RCGA algorithm described in Harpman (2012, Appendix 8) has two parameters in addition to population size (np). These are the reproduction probability parameter, also known as the crossover rate (χ) and the mutation rate parameter (μ). The parameter values used in this research project are shown in Table 5.

As alluded to in Harpman (2012, Appendix 8), there are an amazing variety of reproductive variations in the realm of basic real coded genetic algorithm. The RCGA algorithm used here was restricted to the subset of possibilities wherein two parents produce either one or two offspring. Following implementation of one of the parent selection approaches, the reproduction probability parameter (χ) controls the likelihood the two selected parents will successfully reproduce. Typically, this parameter is chosen in the range of 0.50 to 1.00. A number of authors suggest setting this parameter from 0.90 to 0.95. Low values of the reproductive probability parameter or crossover rate (χ) effectively limit the genetic diversity in the population from one generation to the next. At the extreme, this can diminish the searching capabilities of population leading to a much more rapid and potentially premature convergence. The mutation rate probability parameter (μ) controls the rate of spontaneous genetic mutation in the offspring. Note that random mutations can be fitness enhancing or fitness degrading. This parameter controls the actions of any one of the various mutation schemes which may be employed in the RCGA. While the specifics of these mutation approaches differ in their details, high values of this parameter result in larger injections of genetic diversity in the population, increasing search behavior in the population. For complex or multimodal problems this can lead to a higher probability the global optima will be identified, naturally at the expense of convergence speed. For convex problems, this additional genetic diversity is primarily manifested as increased solution time and expense. A relatively lengthy review of studies on the effects of np, χ and μ on RCGA performance can be found in Haupt and Haupt (2004).

Table 5.—RCGA Parameter Summary

Name	Abbreviation	Range	Setting Used
Population size	np	10 – 2*dim	50
Reproductive probability	χ	0.50 – 1.00	0.90
Mutation probability	μ	0.01 – 0.50	0.02

DE Parameters

The basic DE algorithm described in Harpman (2012, Appendix 9) has two parameters in addition to population size (np). These are the parent scale parameter (F) and the crossover (CR) parameter. The parameter values used in this research project are shown in Table 6.

In the DE algorithm, the offspring or donor vector is constructed from three randomly chosen members of the population scaled by the parameter F. This parameter controls the importance of the parent traits, relative to those of the offspring. High values of F diminish the searching capabilities of population leading to a much more rapid convergence. This can lead to a higher probability of spurious convergence, or convergence at the location of a local optimum. Typically, this parameter is chosen in the range from 0.10 to 1.0. The crossover parameter (CR) controls the probability of crossover. If a randomly generated value exceeds the CR value, the parental trait is passed to the donor individual; otherwise the offspring trait is maintained. Greater values of the CR parameter have the effect of favoring offspring traits in the population, over succeeding generations. Relatively high CR values increase the range of search behavior in the population. For complex or multimodal problems this can lead to a higher probability the global optima will be identified, albeit at the expense of convergence speed. The CR parameter is typically chosen in the range from 0.10 to 1.0.

Table 6.—DE Parameter Summary

Name	Abbreviation	Range	Setting Used
Population size	np	dim – 3*dim	50
Population scale parameter	F	0.10 – 1.00	0.80
Crossover parameter	CR	0.10 – 1.00	0.30

PSO Parameters

The basic PSO algorithm described in Harpman (2012, Appendix 10) has two parameters in addition to population size (np). These are the cognitive weight parameter (c1) and the social weight (c2) parameter. The parameter values used in this research project are shown in Table 7.

The cognitive weight (c1) parameter in the PSO algorithm controls the weight or importance of personal best information found by the individual particle itself, relative to the other members of the swarm. If the value of this parameter is relatively high, more weight or memory is accorded to locations in the search space that the individual particle has personally visited and less weight is given to information provided by the other members of the swarm. As a result, relatively

high c1 values increase the searching behavior of the particle. For complex or multimodal problems this can lead to a higher probability that the global optima will be identified, albeit at the expense of convergence speed. The social weight (c2) parameter in the PSO algorithm controls the weight or importance of social information found by the individual particle itself, relative to the other members of the swarm. The specifics depend on whether the neighborhood or global optimization strategy is employed. If the value of this parameter is relatively high, more weight or memory is accorded to locations in the search space which have been visited (collectively) by other members of the swarm and less weight is given to the particle's own personal best information. As a result, relatively high c2 values reduce the searching behavior of the particle, leading to a much more rapid convergence. For complex or multimodal problems this can lead to a higher probability of spurious convergence, or convergence at the location of a local optimum. For convex problems with a single optima, rapid converge is a desirable characteristic.

Table 7.—PSO Parameter Summary

Name	Abbreviation	Range	Setting Used
Population size	np	20 – 50, dim	50
Cognitive weight[1]	c1	1.0 – 4.0	2.80
Social weight[1]	c2	1.0 – 4.0	1.50

[1] The Clerc (2006) constriction factor, used in this effort, requires c1 + c2 ≥ 4.0.

ABCO Parameters

The constrained artificial bee colony (ABC) optimization algorithm (Karaboga and Akay 2011), the basics of which are described in Appendix 1, has three parameters in addition to population size (np), which corresponds to the number of employed or worker bees. As illustrated in Table 8 these are the limit parameter (Limit), the scout production period (SPP) parameter and finally the modification or perturbation rate (MR) parameter. As described in Karaboga and Akay (2011), the value of two of these additional parameters are based on the number of variables in the problem (dim) and the number of employed bees (np) used for solution of the problem.

The modification rate (MR) parameter controls the generation of new solutions by employed bees. During each iteration, an employed bee undertakes a localized search for new and improved solutions. Potential local solutions are generated in the following fashion. A uniformly distributed random real number (R_j) is first generated. If the value of R_j is greater than or equal to MR, a new value of the variable for dimension j is generated (see Appendix 1 equation 12) and substituted into the solution vector. If the value of R_j is less than MR, the current variable

value for that dimension is retained in the solution vector. For the constrained ABCO algorithm, at least one variable in the new solution vector is always changed. The fitness of this new solution vector is then compared to the current solution using Deb's method (2000) which was described previously. The solution with superior fitness is then retained by the employed bee.

Table 8.—ABCO Parameter Summary

Name	Abbreviation	Range	Setting Used
Number of employed bees[3]	np	10 – 50, dim	50
Modification or perturbation rate	MR	0.40 – 1.00	0.80
Maximum number of iterations allowed without fitness improvement	Limit	various	5
Scout Production Period	SPP	various	10

As the value of the MR parameter is increased, there is a higher probability that one or more values of the original solution vector will be replaced by new values. Conversely, the lower the value of the MR parameter, the lower the probability the value of any particular value in the original solution vector will be replaced by a new value. Higher values of MR result in more searching of the solution space and slower convergence characteristics, and *vice versa*.

The limit parameter (Limit) in the ABC algorithm controls the length of time an employed bee's solution can maintain a solution which is not improving. The greater the value of the Limit parameter, the longer (in terms of the number of iterations) a solution which is not improving will be retained by the employed bee. For complex or multimodal problems this can lead to a higher probability of spurious convergence, or convergence at the location of a local optimum. For a convex problem with a single optima, rapid convergence is a desirable characteristic. When a solution fails to improve by the time the value of the Limit parameter is reached, the employed bee abandons the solution, becomes a scout bee and is randomly assigned a new solution. Smaller values of the limit parameter enhance exploration of the search space and increase the probability that superior solutions will be identified.

[3] Karaboga and Akay (2011) first set the bee colony size (CS). They then set the number of employed bees (np) to ½ CS and the number of onlooker bees to ½ CS. The same approach is followed here. For this reason, a setting of np=20 implies a CS=40.

The scout production period (SPP) parameter controls the period of time (in terms of the number of iterations) when artificial scout bees will be produced. During the SPP period, scouts may be randomly produced whether a non-improving solution equals or exceeds the limit parameter, or not. The scout production process is a diversity enhancing mechanism that allows new and potentially infeasible areas of the search space to be explored.

Variant Selection

Disregarding hybrid approaches (which are discussed elsewhere in this document) a wide range of variations on the basic evolutionary algorithms have been described, and are in use. To reiterate, the term "variants" is used in this document as a general descriptor for these. The number of variants seems to be roughly proportional to the elapsed time since the algorithm was first described and seems limited only by aggregate researcher creativity and the need to differentiate research products for publication.

This effort benefited from a relatively extensive search of the pertinent literature completed previously (S&T Scoping Project ID Number 5992). This component of the study allowed for the admittedly subjective identification of the mainstream algorithm variants. One editorial aside-- several of these algorithm variants are considerably more complex than the underlying algorithms themselves. Some of the mainstream and potentially useful variants are described subsequently and were implemented for this research effort.

RCGA Variants

As a class, the RCGA and GA's exhibit the greatest range of variants on the basic algorithm. Disregarding the hybrid approaches (discussed elsewhere), there are an astonishing number of parent selection approaches, population survival methods, mutation rules and crossover approaches, some of which are amazingly incredibly complex and computationally intensive. Haupt and Haupt (2004), Michelwicz (1996, 2010) and Peltokangas and Sorsa (2008) provide a relatively extensive sampling of these variants.

Parent selection in the RCGA is the method by which two parents are selected from the population for potential reproduction. Three parent selection methods were selected from the literature for use in this research effort. They are the random parent selection method (Random2), the four person tournament method (Tournament4) and the two person tournament approach (Tournament2). Under the Random2 method, one individual is selected systematically from the population as a whole and a second (different) individual is selected randomly from the population. The two selected individuals are then available for potential reproduction (crossover and mutation). The Tournament4 approach selects four different individuals from the population as a whole. Two of these individuals

then compete and the one with the greater fitness becomes a finalist. The two other individuals then compete and the one with the greater fitness enters the finals. The two finalists then compete, and the one with the greater fitness wins and is available for potential reproduction. The Tournement2 approach selects two different individuals from the population as a whole. These individuals then compete and the one with the greater fitness becomes a potential parent.

Crossover is the mechanism by which two potential parents exchange genetic material to create one or more offspring. Crossover in the RCGA context differs considerably from the binary GA case and numerous approaches have been developed to simulate this process. In the context of RCGA, the arithmetic crossover (Arithmetic) approach (Michalewicz 1996), the Laplace crossover (Laplace) approach (Deep and Thakur 2007), the linear crossover (Linear) approach (Wright 1991) and the heuristic crossover (Heuristic) approach (Michalewicz 1996) were implemented for this research effort. The Laplace crossover approach is described in detail in Harpman (2012, Appendix 8). The linear (Linear) crossover approach produces three offspring using an extrapolation approach. An extrapolation weight, often 0.50, is employed. Any variable straying outside the feasible search domain is either censored or the solution is discarded-- the two offspring with the greatest fitness are retained. The uniform arithmetic (Arithmetic) crossover approach is often attributed to Michelwicz (1996). This approach uses a randomly generated value (0, 1) to form a set of weights (α, 1-α) which are then used to create a linear combination of the parent genes. A variant of this approach utilizes a different random value (and hence weight) for each choice variable represented in the parent genetic material. The heuristic crossover approach (Heuristic) was also developed by Michaelwicz (1996) primarily for use in constrained optimization problems. The heuristic approach generates a possible offspring from a randomly weighted differencing of the parent's genetic material, added to the superior parent's existing genetic material. If the offspring lies outside of the feasible domain, a new random weight is generated until a feasible solution is obtained.

Mutation helps to ensure genetic diversity is maintained in the population and some of the mutation variants described in the literature are quite ingenious. For purposes of this research effort three mutation approaches were selected and implemented. These approaches include the Gaussian mutation approach (Gaussian), the nonuniform mutation (Michalewicz 1996) approach (Nonuniform), and the uniform mutation (Uniform) approach (Michalewicz 1996). The nonuniform mutation approach is described in Harpman (2012, Appendix 8) and is not further described here. Under the Gaussian approach, a normally distributed random variable is added to the gene selected for mutation. This approach is relatively simple and effective in many applications although it does require the variance of the distribution to be specified, in some manner. Under the Uniform approach, genes have an equal probability of mutation. A gene selected for mutation is replaced with a uniform random value generated within the feasible search domain. This approach has two advantages. First, it is relatively easy to implement. Second, it executes rapidly.

Survival or recruitment, sometimes also known as replacement, is the process of determining which individuals from the offspring population and the parent population will survive into the next generation. There are a wide variety of recruitment approaches, which have evolved over time (see Reeves 2010 p. 71 for a summary). For purposes of this research effort, three survival approaches were selected from the literature and implemented in code. These are the traditional (Traditional) approach, the Elite_1 approach (Bucknall 2002) and a more general characterization of the elite approach, the elite np (Elite_NP) approach.

The traditional approach to recruitment is fairly straightforward—only the offspring survive into subsequent generations. While easily implemented in code, there is a distinctive logic flaw inherent with this approach. In the traditional approach there is a probability the individual with the highest fitness will be eliminated from the gene pool, slowing the evolutionary process and the search for an optima.

The Elite_1 approach preserves the genetic material from the fittest individual in the gene pool from one generation to the next. In the Elite_1 recruitment approach, the parents are ranked from highest fitness to lowest fitness and the offspring are ranked from highest fitness to lowest. The parent individual with the highest fitness (the Elite_1) replaces the lowest ranked offspring, provided it is of superior fitness. The remaining offspring and the Elite_1 individual, survive into the next generation.

There are many potential variations on the elitism approach. Conceptually, the retained elite fraction could vary all the way up to NP (here assuming a constant population size is maintained). For purposes of this project, the Elite_NP approach was employed. Under this approach all of the parent and offspring individuals are pooled and then sorted by fitness. The most-fit NP individuals from the pool are then retained and survive into the next generation, the less fit individuals are removed from the gene pool. This approach greatly increases convergence speed, often dramatically. Unfortunately, the genetic diversity of the population diminishes as well and the likelihood of spurious convergence, or convergence at a local optimal point, increases.

DE Variants

Like the other evolutionary algorithms explored here, there are a number of variants on the basic DE algorithm. Disregarding the hybrid approaches (discussed elsewhere), there are a large number of mutation rules and crossover approaches, some of which are amazingly ingenious. Many of these build upon the seminal DE paper (Price and Storn 1995, 1997) which described 23 crossover and mutation combinations. Over time, a shorthand approach for describing and categorizing the more mundane of these variants has evolved. The notation DE/x/y/z is often used for this purpose. In this notation, x is used to specify the vector to be mutated which can be "Rand" (a randomly chosen member of the

population) or "Best" (the member of the population with the current best fitness), y represents the number of difference vectors used, and, z denotes the type of crossover scheme employed. The most common crossover variant is the "Bin" or binary crossover approach.

For purposes of this research effort, six different crossover and mutation approaches were selected. These include the originally described DE/RAND/1/BIN and the DE/BEST/1/BIN approaches, but also include some of the more promising and exotic approaches such as the random scale factor (DERANDSF) approach, the trigonometric (TRIGON) approach, the time varying scale factor approach (DETVSF) and the self adaptive (SELFADAPT) approach (Brest et al 2006 version). These variants were sufficiently represented in the mainstream literature to warrant further investigation.

PSO Variants

Many examples of PSO variants can be found in the literature, the majority of which are reviewed in Valle et al (2008). Disregarding hybrid approaches (discussed elsewhere in this document) the two enduring variants appear to be the application of global or neighborhood optimization strategies. In the global optimization strategy, crossover is a linear combination of the best fitness value found by any of the members of the swarm (globally) and a particle's personal best fitness. This optimization approach results in faster convergence but also decreases searching behavior and increases the likelihood of spurious convergence or identification of a local, rather than global, optimal point. This optimization strategy should be distinguished from the neighborhood (or local) optimization strategy. In the neighborhood optimization strategy, crossover is a linear combination of the best fitness location identified by any of the members of a particle's neighborhood and the particle's own personal best fitness value. Figure 9 illustrates a globally connected swarm of np=8 (Panel A) and an np=8 swarm with a 3-member neighborhood structure or topology (Panel B).

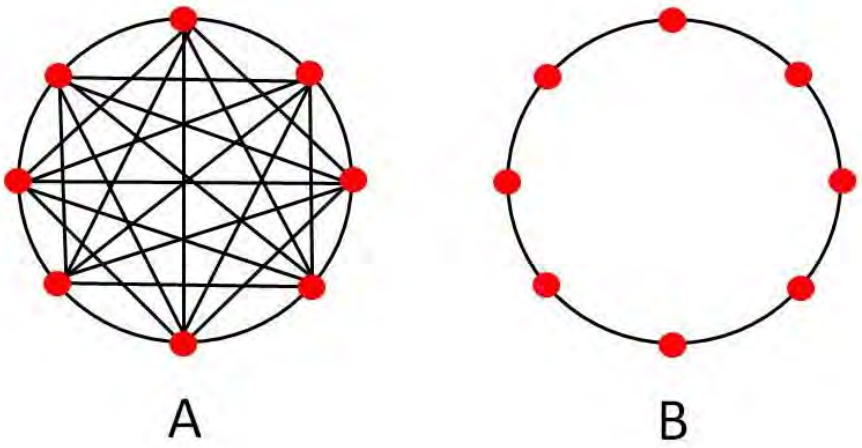

Figure 9.—Globally connected (A) and 3-neighbor (B) swarms.

The use of a neighborhood structure serves to limit the information about the search space available to any single member of the swarm. Neighborhoods can, and are, constructed in a variety of shapes or topologies and follow an amazingly creative set of behavioral rules (see Kennedy and Mendes (2002) for some of the details). For purposes of this research, a star-type neighborhood topology, limited to five total members (including the particle itself) was employed. These 5-member neighborhoods limit the overlap or interconnection in the swarm. With each succeeding generation, information about the location of potential optima effuses from neighbor to neighbor, and from neighborhood to neighborhood within the swarm. This approach results in enhanced searching behavior, slower convergence times, and it reduces the probability of spurious convergence and convergence on local optima within the search space. The 5-member star-type neighborhood configuration proved to be relatively straightforward to implement in code and highly effective in application.

Following the literature review component of this research effort, Clerc's constriction coefficient (2006) was selected for use in the PSO algorithm. This PSO variant is described more completely in Harpman (2012, Appendix 10).

ABCO Variants

The basic artificial bee colony (ABC) optimization algorithm described in Appendix 1 of this document is relatively new. While some variants on the continuous algorithm have been reported in the literature, the number and complexities of these are few, relative to more mature evolutionary algorithms. The majority of these variants are reviewed in Karaboga et al (2012), Kauer and Goval (2011) and Teodorivic, Davidovic and Selmic (2011). Some potentially useful variants are also described in Subotic (2012), Alzaqebah and Abdullah (2011) and Tuba, Bacanin and Stanarevic (2012).

Disregarding hybrid approaches (such as Abraham, Jatoth and Rajasekhar 2012) we focus the discussion on four modifications proposed by Mezura-Montes and Cetina-Dominguez (2012). In their paper, Mezura-Montes and Cetina-Dominguez (2012) introduce and test four variants of the constrained ABCO algorithm. They propose modifications to the selection mechanism, the scout bee operator and the behavior of the algorithm in the presence of equality and boundary constraints. Three out of four of these variants were implemented and used in this research project. The ABCO results reported later in this document reflect the performance improvements, if any, afforded by these variants.

In the basic ABCO algorithm described in Appendix 1 information is shared between the employed bees and the onlooker (unemployed) bees by a waggle dance. This mechanism is emulated by a fitness proportional sharing mechanism as shown in Appendix 1 equation (14). Mezura-Montes and Cetina-Dominguez (2012) modify this approach by using tournament selection, which is also used in the Real Coded Genetic Algorithm (RCGA). Under the tournament2 approach, two different individuals are randomly selected from the population as a whole. These individuals then compete and the one with the greater fitness wins, or is selected. Again, the individual with greater fitness is identified by following Deb's rules (Deb 2000), which were described previously in this document.

In constrained optimization problems, equality constraints present a special challenge to obtaining a successful solution. In the original ABCO algorithm, equality constraints are rewritten as inequality constraints which are satisfied to a specified violation tolerance level (vtol) which is statically set. Mezura-Montes and Cetina-Dominguez (2012) introduce a mechanism for dynamically varying the violation tolerance level for satisfying equality constraints. They introduce the use of a dynamic feasibility tolerance parameter which is shown in equation (7).

(7) $$\varepsilon(g+1) = \frac{\varepsilon(g)}{dec}$$

Where: ε = value of the tolerance level
g = iteration number
dec = fixed rate of decrease (dec>1.00)

As illustrated in equation (7), using this dynamically varying tolerance level, the initial iterations have a relatively large feasible region which satisfies the equality constraint. As the iterative process proceeds, the feasible region diminishes, shrinking around the true equality feasible region. This dynamic mechanism is said to admit more potentially optimal solutions and facilitate a relatively more rapid convergence on the true constrained optimum.

As described elsewhere in this document and in Harpman (2012), a mixed penalty and repair system was developed to enforce equality and inequality constraints. The total release volume constraint is characterized as an equality constraint. A relatively sophisticated release volume repair function was implemented and is used in all of the other EAs examined here. To leverage this existing work and ensure comparability between the algorithms, this volume repair function (and the remaining system of penalty functions) is also employed for the ABCO algorithm explored in this research effort.

The scout bees explore the search space and identify new, potentially feasible and improved solutions. Employed bee solutions which do not improve in a pre-set number of iterations are replaced by scout bee solutions. The scout bees use a random mechanism to identify new solutions. In constrained optimization problems the feasible region may be relatively small and many of these randomly generated solutions may be outside of the feasible region. Mezura-Montes and Cetina-Dominguez (2012) modify this approach in a complex but potentially more information efficient manner. They introduce the "smart flight operator" represented in equation (8).

(8) $$v_{i,j} = x_{i,j} + rand(x_{k,j} - x_{i,j}) + (1 - rand) * (x_{B,j} - x_{i,j})$$

where:
$v_{i,j}$ = new solution vector
rand = uniform random value ($0 \leq rand \leq 1$)
$x_{i,g}$ = current solution vector
$x_{k,j}$ = randomly chosen vector
$x_{B,j}$ = global best solution vector

Equation (8) linearly combines information about the solution to be replaced, the location of a new randomly chosen solution value and the value of the global best solution. This approach is very similar to that used in particle swarm optimization (PSO). The authors maintain it may lead to a feasible solution or at least, an infeasible solution closer to the feasible region. This smart flight operator was implemented, tested and incorporated into the ABCO algorithm used in this research project.

Lastly, Mezura-Montes and Cetina-Dominguez (2012) introduce a different method for handling boundary constraint violations. In the basic ABCO algorithm described in Appendix 1 both the scout bees and the employed bees may generate solutions outside of the feasible range. Should this occur, values outside of the allowed feasible range are reset to either their upper limit (U_j) or their lower limit (L_j), as appropriate. Mezura-Montes and Cetina-Dominguez (2012) modify this simple procedure by using a similar, but more informed transformation technique represented by equation (9).

$$v_{i,j} = \begin{cases} 2 * L_j - v_{i,j}, & if\ v_{i,j} < L_j \\ 2 * U_j - v_{i,j}, & if\ v_{i,j} > U_j \\ v_{i,j}, & otherwise \end{cases} \quad (9)$$

where: $v_{i,j}$ = solution vector
U_j = upper bound of variable j
L_j = lower bound of variable j

Using this approach, if the randomly generated solution is less than the lower bound value, it is reset to 2*L minus the solution. If the randomly generated solution exceeds the upper permissible value, it is reset to 2*U minus the solution. Lastly, if the randomly generated solution is within feasible bounds, it is not transformed.

The boundary constraint handling method described by Mezura-Montes and Cetina-Dominguez (2012) was implemented, tested and incorporated into the ABCO algorithm used in this research. One incremental improvement was made to their approach. The approach described by the authors does not necessarily generate a value in the feasible domain. As a failsafe mechanism the approach used by Mezura-Montes and Cetina-Dominguez (2012) was augmented with traditional boundary constraint enforcement code. If a solution outside the feasible search space is generated by equation (9), it is reset to either the upper or lower boundary of the space.

Mezura-Montes and Cetina-Dominguez (2012) report the four variant modifications discussed here result in enhanced performance, relative to the original ABCO algorithm. They state this is particularly evident in constrained problems with equality constraints characterized by small feasible solution regions. Three out of four of these suggested modifications were incorporated into this effort.

Development Process

This research project required an extensive behind the scenes software development effort. For the most part, the evolutionary algorithms examined in this research effort are rather new and certainly not commercially available. A relatively large-scale and time consuming development effort was required to make them operational.

Development Platform

The Embarcadero Rapid Application Development (RAD) Studio XE2, an object oriented rapid application development (RAD) environment was employed for coding the majority of programs used in the Phase 2 research project. Phase 1 of this project and the early stages of Phase 2 utilized the Borland Developer Studio (BDS) 2006 platform. The change in software development tools from BDS 2006 to RAD XE2 was necessitated by Reclamation's move to a new operating system.

In January 2013, the Bureau of Reclamation officially migrated to the Microsoft Windows 7 operating system. Initial indications from Embarcadero (which has now absorbed the original manufacturer, Borland Inc.) and third party users (see, for example http://www.drbob42.com/examines/examin84.htm) were that Borland Delphi Studio 2006 could successfully be deployed on the Windows 7 operating system. Although in general this may have been true, apparently this outcome did not extend to secure, managed systems. Four attempts were made to install Borland Developer Studio 2006 on the Department of the Interior's variant of the secure Windows 7 operating system. All of these efforts proved to be incredibly time-consuming-- and entirely unsuccessful.

The Embarcadero Rapid Application Development (RAD) Studio XE2, a newer incarnation of this development tool, was purchased and successfully installed on the Windows 7 operating system. Additional time was devoted to identifying a third-party visual component for sizing and scaling application screens in this new environment.

The Embarcadero RAD Studio XE2 development environment is designed for use on Microsoft Windows 32 bit and 64 bit operating systems, such as Windows 7 and Windows 8. Although the development environment has web, web application, C, C++ and Apple OS X capabilities, the Delphi language (for Windows) was used throughout this project. Delphi is an object oriented language which evolved from the Turbo PASCAL language, popular in the early 1980's.

The Embarcadero RAD Studio XE2 compiler produces native Windows 32 bit and 64 bit executable code. This environment eases the development of Windows based graphical user interfaces while allowing full code control. Of particular advantage for this project, the development environment includes visual component libraries (VCLs) for advanced graphics and database integration. These VCLs along with others for printer and device control, disk file operations and interfacing with the windows environment are implemented with "drag and drop" functionality in a fully visual integrated development environment. This greatly streamlines the development of Windows based applications allowing the researcher/developer to devote their resources to code development. The integrated graphics capabilities were a major consideration in the decision to deploy this platform.

Other development platforms were reviewed for possible use in this project. These platforms included MATLAB (www.mathworks.com), a commercially

available package widely used in engineering applications, the open source European equivalent, SCILAB (www.scilab.org) and the open source general purpose statistical package, R (www.r-project.org). However, none of these platforms appeared to offer the ease of development, stability and integrated graphics capabilities required for this effort.

Three Stages of Development

Following the selection of candidate algorithms, a three stage development process was undertaken. First, the algorithm was coded and tested on three unconstrained test problems. Second, the algorithm was coded and tested on the hydropower unit dispatch problem. Third, a testing environment was developed for each algorithm.

Stage 1—Development with Test Problems.

Working from pseudo-code, flow charts, verbal descriptions in journal articles, code snippets and in some rare cases, translating from purportedly functional C source code, the selected algorithms were coded in Delphi, debugged and brought to an operating state. The three unconstrained three dimensional (3-D) test problems described in Appendix 4 were used during this stage to debug, and more importantly, test the functioning and solution behavior of the coded algorithms.

A graphical user interface (GUI) was developed for each application. These GUI's (naturally) share a number of common features and functionality. Shared features include a tabbed page for selecting an initialization strategy, a tabbed page for selecting a convergence strategy and convergence tolerance and a tabbed page for controlling visualization.

Each GUI also has an Algorithm tabbed page which is customized for each algorithm. This customized tabbed page allows for easy user control of parameters specific to the algorithm and allows different variations of each algorithm to be selected. For example, the Algorithm tabbed page for the DE algorithm allows the user to select from a list of Mutation strategies, select the number of individuals in the population (np), set the value of the scale parameter (F) and select the value of the crossover (CR) parameter. In contrast, the RCGEN algorithm tabbed page allows the user to set the number of individuals in the population (np), select a parent selection strategy, select a crossover approach, select a mutation strategy and select a recruitment approach.

All of the applications share a common output GUI configuration, shown in Figure 10. Each application has a numerical output window and a graphical output window. The latter allows for real-time visualization of solution progress, a feature which has proven to be invaluable.

The behavior of the algorithms using different parameter settings and optional variants was observed both numerically and visually by judicious application of

the integrated graphics capability. Figure 10 illustrates the graphical output screen of the RCGA program at iteration 6 during a solution of the Alpine function. This figure shows the plan view of this relatively complex function (see Appendix 4 for further details about this and other test functions). In the figure, the blue diamonds illustrate the (x,y) locations for each of the np=40 individuals in the population. The single red diamond located in the upper right-hand quadrant indicates the location of the optimal solution in the bounded search space.

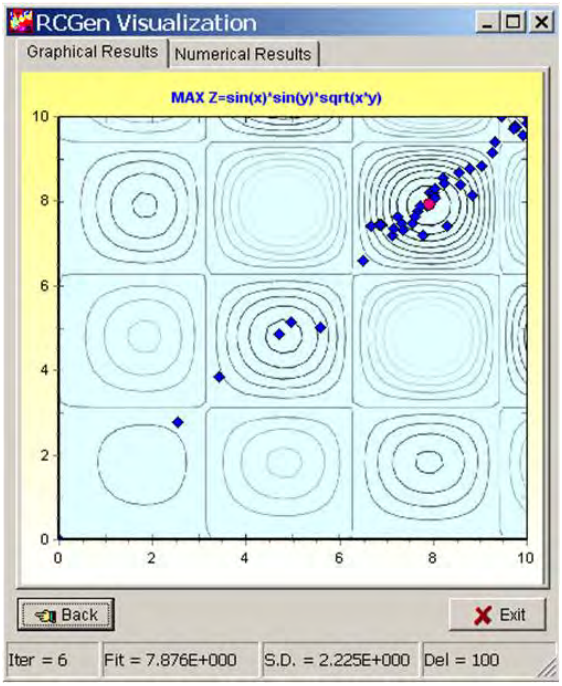

Figure 10.—RCGEN Program Solving the Alpine Function.

The integrated graphics allows the researcher to observe the solution behavior in real-time while simultaneously monitoring the algorithm's numerical progress toward a solution. Progress metrics are written to the status bar at the bottom of graphics window. As reported in the status bar, shown in Figure 10, this plot is for the sixth iteration, the most fit individual in the population has a fitness (Fit) of 7.786E+000, the standard deviation (SD) of population fitness is 2.225E+000 and the visual delay (Del) is set to 100 milliseconds.

Implementation of these evolutionary algorithms involved overcoming a number of technical travails. This included selecting and developing a random (pseudorandom) generator, the use of low discrepancy sequences, the development of appropriate convergence or stopping criteria and the development and application of constraint and constraint handling methods.

Stage 2—Unit Dispatch Problem

Working from the code base developed in Stage 1 of the development process, the evolutionary algorithms were adapted for solution of the hydropower unit dispatch problem. The four test unit dispatch problems described in Appendix 6 were used during this stage to debug and complete initial tests on the coded algorithms. The unit dispatch problem is a piecewise, discontinuous constrained optimization problem and accommodating this problem required a further and rather extensive coding effort in its own right.

The graphical user interface (GUI) developed in Stage 1 of the development effort was modified to accommodate the four unit dispatch test problems. Added GUI features included a tabbed page for selecting one of the four test unit dispatch problems, a tabbed page for designating which, if any, of the generation units would be devoted to condensing and a page to indicate the amount of water scheduled for release. Additionally, a tabbed page was added to allow for more detailed monitoring of the numeric progress towards a solution.

All of the common output GUI's were modified to better suit the unit dispatch problem. The graphical output for this application was modified, as shown in Figure 11, to illustrate the minimum and maximum release constraints and display the optimal hourly pattern of generation and release, unit by unit. This visual output screen is particularly useful because it illustrates which units are condensing, if any, and the location of rough zones for each unit, if any. The numerical output window was revised to show the hourly details of the optimal solution for this problem. An additional numeric output window was added to record selected intermediate output metrics for each iteration (or generation) as the algorithm evolved towards a solution. This proved to be an invaluable debugging aid. Figure 11 illustrates the graphical output for a default solution of the 4-unit dispatch problem with rough zones

.

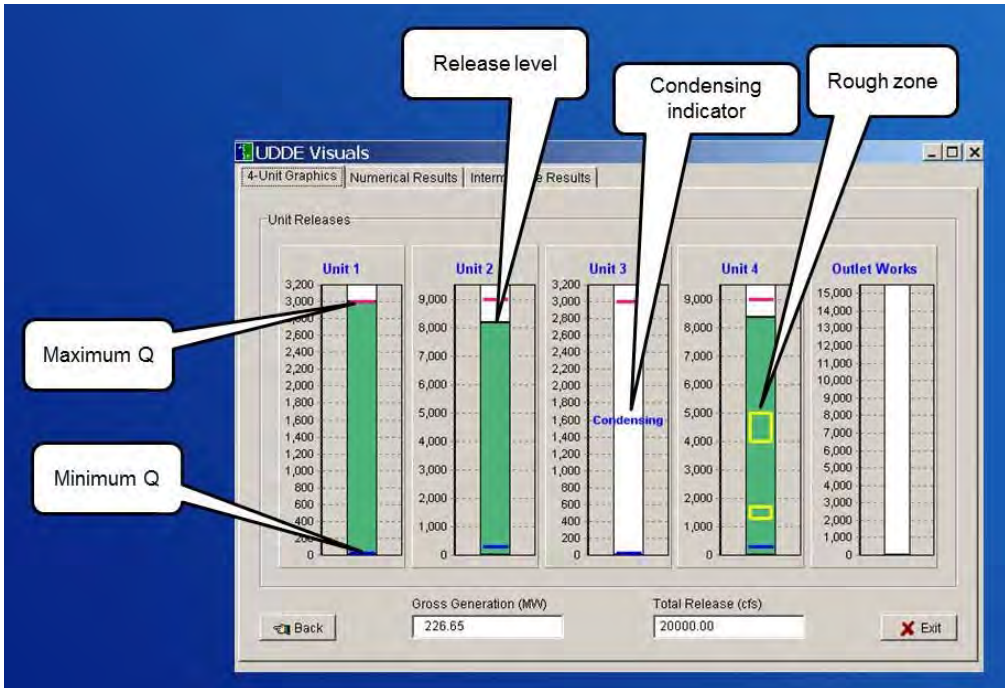

Figure 11.—HDDE Solution to Test Problem 4.

Stage 3—Testing Environment

The purpose of Stage 3—development of a testing environment, was to construct a framework for the unattended replication of experiments while saving success metrics, performance measures, numerical outcomes and other summary data for subsequent statistical analysis. As described previously in this document, evolutionary algorithms are stochastic in nature. For any given set of starting values, an algorithm may achieve a different, slightly different, or vastly different solution. Or, it may fail altogether. This range of potential outcomes arises because of (a) the initialization approach employed, (b) the random underpinnings of their solution behaviors, and (c) the approach implemented to detect convergence on a solution. Consequently, a single successful solution, while indicative, is by no means conclusive evidence of the successful application of one of these algorithms. In the context of evolutionary algorithms, replicated trials followed by statistical analysis are required to support even minimal conclusions about their suitability for a specific class of problem.

Stage 3 development required using the code base developed in Stage 2 of the development process, and adding additional code to allow for repeatedly running the algorithm and recording salient success and performance measures for each run. Relative to the development effort expended in the first two stages of development, this was readily accomplished requiring only minor modifications to both the input and output GUI's and limited (additional) code development. The testing environment developed in this, the final Stage of the development

effort, was utilized to produce the replicated experimental results described subsequently in this document.

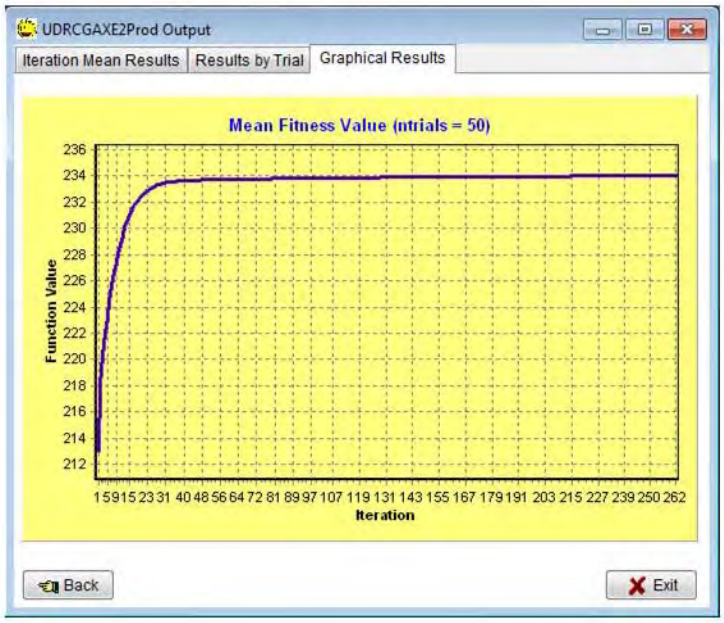

Figure 12.—Test Environment XE2 Graphical Output.

Selected Experiments

Given the number of parameters, options, algorithm variants, input vectors and problem features described in this report, a very large number of experiments could be undertaken. While a comprehensive effort would surely be a valuable contribution to the state of scientific knowledge, resource limitations dictated that only selected experiments be completed and reported in this document. The experiments which are described here were selected primarily to provide insights about the applicability of EA's to the unit dispatch problem, their performance and the factors which might influence this decision.

Box and Whisker plots are used to graphically display the results of many replicated experiments undertaken for this project. These plots summarize the relevant statistical details of replicated experiments and facilitate comparisons across treatments. Box and Whisker plots show the mean, median (50^{th} percentile), interquartile range (25^{th} to 75^{th} percentile) and extent of the empirical distribution which lies between the 5^{th} and the 95^{th} percentile. A useful review of Box and Whisker plots can be found in Appendix 9. With a nod towards

readability, the detailed numerical results of many of the experiments are reported in the Appendices.

Initialization Approaches

Review of the pertinent literature revealed a number of efforts which reported systematic differences in EA performance resulting from initialization method. The literature on this topic was both extensive and conclusive—the use of the uniform random approach for initialization was judged to be inferior to other approaches. Based on this body of literature, considerable researcher effort was allocated to developing, testing and applying promising alternative initialization approaches to the hydropower problems examined in this research effort. This required the testing and development of four low discrepancy sequences including the Niederieter, the Habor, the Weyl/Torus and the Halton (see Harpman 2012 Appendix 13 for additional explanation) as well as the opposition based learning (OBL) method.

After coding and validating the functioning of these different initialization approaches, a systematic set of performance experiments was undertaken. For purposes of the replicated initialization experiments described here, 50 trials were undertaken on test problem 4, the 4-unit dispatch problem. The summary characteristics of test problem 4 can be found in Table 15 and are further described in Appendix 6. To ensure a valid comparison across algorithms, the population (swarm) size was set at 50 individuals for all of the evolutionary algorithms (np=50). It should be noted the performance of some of the EA's, such as PSO and RCGA may be disadvantaged by setting the population size to this level for this comparatively small dimension problem. A common stopping rule, the Elite_SD rule (with tol=1.0e-04), was employed for all of the replicated experiments carried out on initialization approaches.

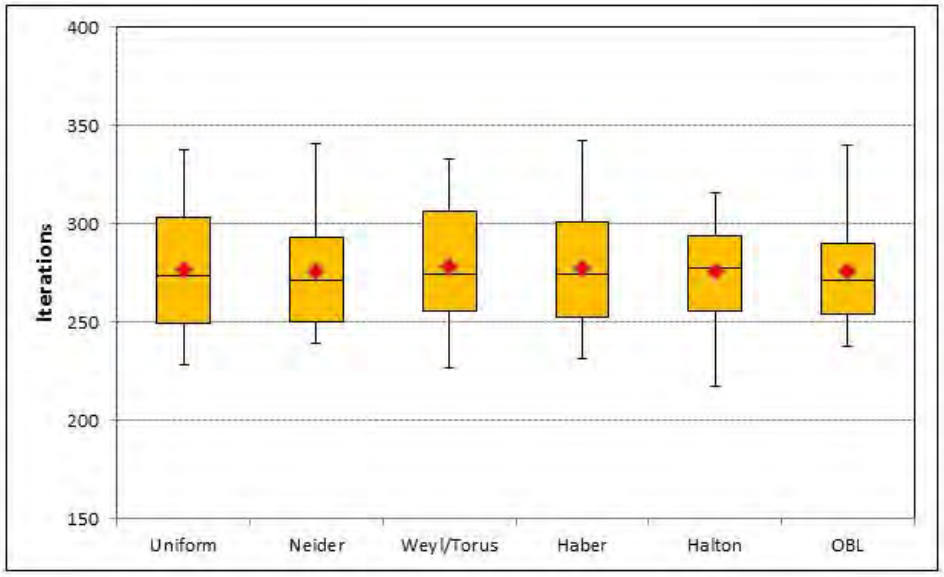

Figure 13.—ABCO Initialization Approaches.

The results for the ABCO algorithm are shown in both Figure 13 and Table 9. These outcomes are both unremarkable and unexpected. For the unit dispatch problem, there appears to be no discernible difference between the uniform random initialization approach and any of the other approaches. While this result appears to contradict the results reported in many earlier studies, none of the preceding studies focused on this particular type of constrained optimization problem.

Table 9.—ABCO Initialization Results

	Uniform	Neider	Weyl/Torus	Haber	Halton	OBL
mean	277.1	275.7	277.9	277.3	275.6	276.3
std. dev	35.4	33.0	34.9	33.1	33.3	31.9
minimum	217.0	212.0	202.0	198.0	216.0	227.0
05th perc	228.4	238.9	227.0	231.0	217.4	237.5
25th perc	249.0	249.8	255.3	252.5	255.3	254.0
median	273.5	271.0	274.0	274.5	277.5	271.0
75th perc	303.0	293.5	306.8	300.8	294.0	289.8
95th perc	338.1	341.1	333.1	342.2	315.7	340.4
maximum	359.0	362.0	355.0	353.0	380.0	363.0

The results for the DE, PSO and RCGA algorithms are very similar to those shown in Figure 13 and Table 9. The detailed experimental results for these algorithms are reported in Appendix 10.

Convergence Behavior

The unit dispatch problem described earlier in the text and discussed more fully in Appendix 6 is a piece-wise discontinuous constrained optimization problem. This problem was strategically constructed expressly to facilitate comparisons between algorithms and to facilitate the other experiments described here. For purposes of the replicated performance experiment described here, 50 trials were undertaken on problem 4, the 4-unit dispatch test problem. To facilitate comparisons, the population (swarm) size was set at 50 individuals (np=50) for all of the evolutionary algorithms. Certain EA's, such as PSO and RCGA may be disadvantaged by setting the population size to this level for this comparatively small problem. The Elite_SD stopping rule with the elite fraction set to 0.90 was employed for all of the evolutionary algorithms with the convergence tolerance set at 1.0e-04. The parameter settings chosen produce a water release of 10,000 af (to two decimal digits of precision) for all the evolutionary algorithms, again facilitating this performance comparison.

Figure 14 compares the convergence behavior of all of the EA's for the first 50 iterations. As shown in this plot, the EA's are able to very quickly identify the region containing the optimum point, illustrating their relative strength in global search. However, all four of the EA's require a number of additional iterations to converge on the optimal point, illustrating their relatively poor local search capabilities.

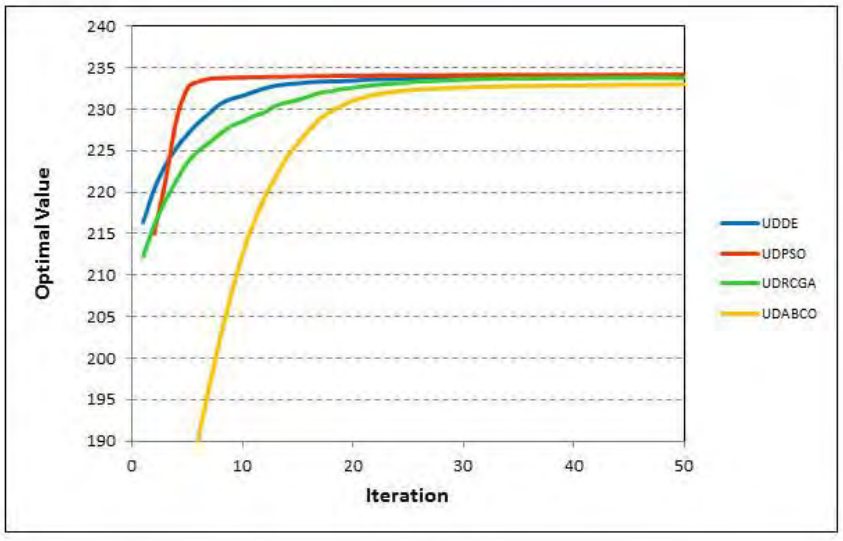

Figure 14.—EA Convergence Behavior

As shown in Figure 14, the PSO algorithm reaches the vicinity of the optimum more rapidly than the other three algorithms. The DE and the RCGA algorithms have similar convergence characteristics and the ABCO algorithm exhibits a more gradual approach to the optimum. After locating the approximate region of the optimal point, all four algorithms slowly converge on the optimum.

Table 10 summarizes the convergence results for each of these algorithms for the SD_Elite stopping rule. Over the 50 trials, the EA's are able to identify essentially the same mean solution for dispatch test problem 4. For this problem, the DE algorithm is the most efficient, the PSO and the ABCO algorithms are very similar and the RCGA algorithm is considerably slower.

Table 10.—Convergence Performance

Algorithm	Mean Number of Iterations	Mean CPU time (msec)
DE	287	32.9
PSO	207	67.1
RCGA	2526	207.1
ABCO	274	64.78

Stopping Rules

Consistent with the philosophy of evolutionary programs, a convergence or stopping rule should utilize the fitness information available for each iteration, be simple and effective. In keeping with this theme, one approach is to use the population mean and standard deviation for detecting convergence. Operationally, the mean and/or standard deviation of the solutions found by all of the individuals in the swarm or population are calculated for each iteration. When these metrics change by less than a pre-set tolerance, or fall within an acceptable tolerance, the algorithm has converged on a solution. The advantage of this method is that it is relatively easy to implement, is problem and scale invariant and is brutally effective. Arguably, this approach may be overly conservative and computationally inefficient often requiring an extensive number of iterations for all of the members of a swarm or population to converge on the optimal point.

Zielinski et al (2006), Zielinski and Laur (2007) and Zielinski and Laur (2008) explore the subject of convergence rules for particle swarm optimization (PSO) and differential evolution (DE). They examine single objective problems with different dimensions using varying population sizes. In aggregate, the authors systematically explored the performance of a suite of approaches, some of which were quite esoteric. They recommend two methods for use with PSO and two

methods for use with DE. They suggest a variant of the standard deviation approach be examined more fully in future research efforts.

Motivated by the work of Zielinski and Laur (2008), two additional convergence criteria were developed and investigated in this research effort. These were the elite mean (Elite_Mean) and elite standard deviation (Elite_SD) approaches. These two approaches are based on the observation that one or more members of the swarm or population will identify the optimal point well before the other members of their cohort. We will call the portion of the population which converges rapidly, the "elites." At each iteration, uninformed members of the cohort will continue to search in the solution space, sometimes far from the optimal point. It can, and often does, require many additional iterations for all of the members of a swarm or population to converge on the optimal point, previously discovered by a few individuals. Observations made in the early phases of this project suggest that a disproportionately large number of iterations are required to produce convergence in the last decile of the swarm or population.

The elite mean and elite standard deviation stopping rules are based on the empirical observation that a subset of particles (the elites) converge very rapidly and a subset of the remaining individuals converge extremely slowly. To take advantage of this, calculation of the elite mean and elite standard deviation convergence metrics are based solely on the behavior of the elite or best performing particles. The behavior of the lower performing individuals, which potentially could take many more iterations to converge, is ignored.

Identification of the elite members of the swarm or population is, of course, somewhat problematic for the purposes described. Since we do not know *a priori* the optima for a given problem, we cannot know with certainty if say, the two most fit particles in iteration number 561 have converged on the solution, or not. If the elite proportion of the population were defined as the most-fit 5%, the potential for spurious convergence may be quite high. Alternatively, defining the elite proportion of the swarm as the most-fit 99% may result in significant and unwarranted computational cost. This choice represents a fundamental analysis trade-off which is to some extent arbitrary, but is surely problem dependent.

During this research some exploration of this trade-off was undertaken. This exploration could not be described as either comprehensive or conclusive. However, it was sufficiently extensive to make some inferences about the application of these stopping rules to the types of problems examined here. After some experimentation, the elite proportion of the population was defined as the most fit 90% of the population or swarm. By definition, the individuals classified as elites varied dynamically from one iteration to the next. Using this definition for the elites resulted in excellent computational performance with very little likelihood for spurious or premature convergence. These stopping rules are relatively easy to implement, do not add extensive computational overhead and proved to be very effective for the types of optimization problems we examined.

For purposes of the replicated stopping rule experiments, 50 trials were undertaken on problem 4, the 4-unit test problem with rough zones (described in further in Appendix 6). To ensure a valid comparison of the different stopping rules, the population (swarm) size was set at 50 individuals for all of the evolutionary algorithms (np=50). It should be noted that some of the EA's, such as PSO and RCGA may be disadvantaged by setting the population size to this level for this comparatively small problem. For applicable "elite" approaches, the elite fraction was set to 0.90. In all cases the convergence tolerance (ctol) was set at 1.0e-04.

Figure 15 shown below compares the performance of four different stopping rules applied to solution of the same problem. It compares the mean central processing unit (CPU) time, measured in milliseconds (msec), required for convergence over 50 trials between the maximum iteration approach (maximum iterations = 5000), the population standard deviation (Pop_SD) approach, the elite standard deviation (Elite_SD) approach and the elite mean (Elite_Mean) approach.

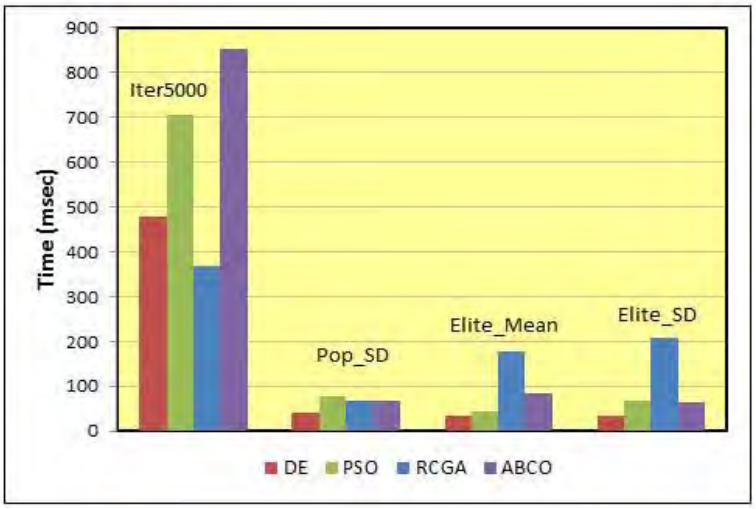

Figure 15.—Results of Stopping Rule Experiments.

As shown in this figure, there is a large difference between the mean computational time required to achieve convergence, when convergence is specified as completing 5000 iterations, and the CPU time required by the three stopping rules which intelligently monitor the progress of the calculation metrics (Pop_SD, Elite_Mean and Elite_SD). Although not reported here, the mean precision of the solutions at convergence is very similar. Over the course of repeated calculations and when the dimensionality of the problem increases, the reduced time necessary to achieve convergence is a substantial advantage to the researcher. There are discernable differences between the CPU time required for convergence when Pop_SD, Elite_Mean and Elite_SD convergence criteria are employed. Potentially, there may be computational advantages to the use of the

Elite_Mean approach, however no statistical analysis was undertaken to explore this possibility further.

Problem Size

In Phase 1 of this research project, an increase in solution times was observed as the size of the constrained optimization problem increased. Similar increases in the solution times are the norm for calculus based optimization approaches. A systematic investigation of this (apparent) performance degradation seemed warranted.

A priori, the influence of increasing the dimensionality of the optimization problem seemed relatively easy to foresee. Increases in the problem size are expected to increase the number of unknown variables, the size of the storage vectors needed, the time to manipulate those vectors and the computational cost of evaluating the fitness function.

For purposes of the replicated stopping rule experiments, 50 trials were conducted on dispatch test problem 1, the 2-unit problem with no rough zones, and problem 3, the 4-unit problem with no rough zones. For these experiments, the population size (np) was set to 50 and the Elite_SD stopping rule was employed with the elite fraction set to 0.90. In all cases the convergence tolerance (ctol) was set at 1.0e-04.

The detailed results of this experiment are reported in tabular form in Appendix 12. Figure 16 shown below summarizes the results of this experiment. This figure compares the mean central processing unit (CPU) time, measured in milliseconds (msec), required for convergence over the 50 trials.

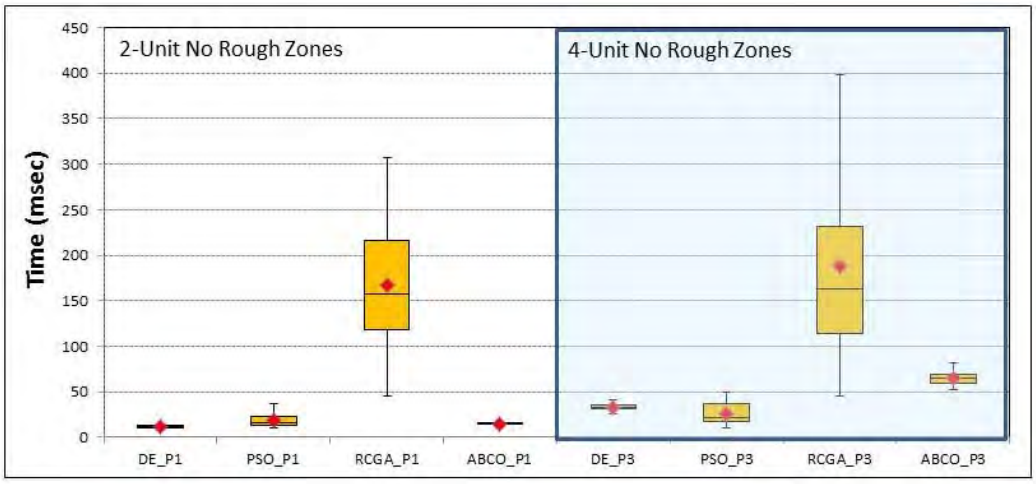

Figure 16.—Results of Problem Size Experiment.

As shown in Figure 16, there is generally a substantial increase in convergence times, across all of the EA's, when the size (dimensions) of the problems are increased from 2 to 4 units (a 2-fold increase in dimension). For most of the EA's, the associated increase in convergence time is much greater than proportional to the increase in problem size. For the RCGA algorithm, the results are less clear, although the variance of the convergence time increases greatly.

Rough Zone Constraints

During the development process, an increase in solution times was observed when there were binding constraints (other than the total release constraint, which is always binding). This was not necessarily expected by the research team since calculus based optimization approaches do not, apparently at least, suffer as much from this phenomenon. A systematic investigation of this (apparent) performance degradation seemed warranted.

As described earlier in this report, a mixed system of penalty functions and repair methods was employed for all of the EA's examined in this project. Some of the repair approaches are quite involved. More frequent and intensive calls to these penalty and repair subroutines are likely to result in longer solution times.

The potential effects of a binding rough zone constraint was systematically explored. Based on the results of previous experiments, it was thought there could be an interaction effect between the binding constraints although this possibility was not formally explored.

For purposes of the constraint experiments, 50 trials were conducted using dispatch test problems 1 and 2. Test problem 1 is a 2-unit problem with no rough zones. Test problem 2 is a 2-unit problem with rough zones. The plant release (Q) was set to 7500 cfs. In the absence of rough zones, the optimal release is 3,000 cfs in unit 1 and 4,500 cfs in unit 2. The optimal release from unit 2, without considering rough zones, is within the range of the rough zone for this unit for test problem 2, which ranges from 4000 to 5000 cfs. Thus for test problem 2, the rough zone constraint is binding. For these experiments, the population size (np) was set at 50 for all of the algorithms as shown in Tables 5 through 8 found earlier in the text. The Elite_SD stopping rule was employed with the elite fraction set to 0.90. In all experiments the convergence tolerance (ctol) was set at 1.0e-04.

The detailed results of these experiments are reported in tabular form in Appendix 13. Figure 17 neatly summarizes the results of these experiments, using Box and Whisker plots. This figure compares the mean central processing unit (CPU) time, measured in milliseconds, required for convergence over the 50 trials.

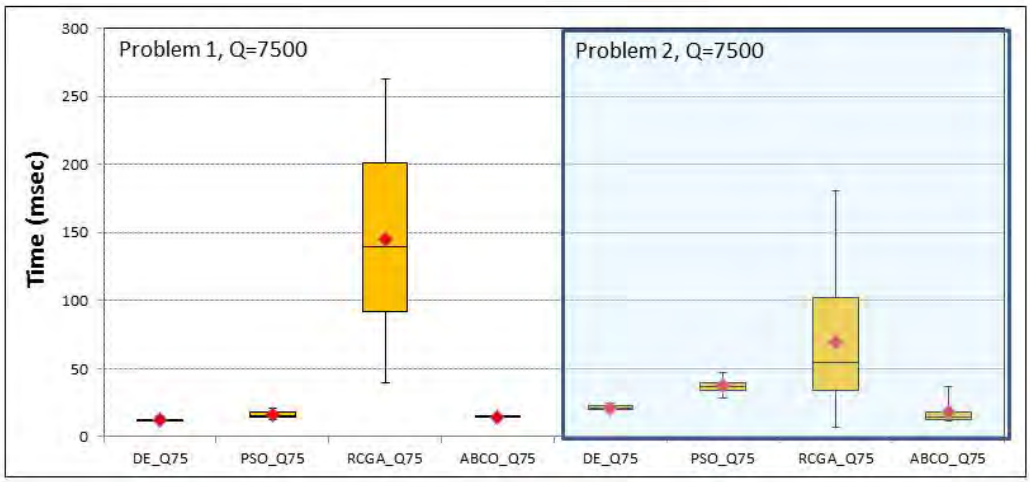

Figure 17.—Convergence with Rough Zone Constraint

As shown in Figure 17, when the maximum release constraint is set to 7,500 cfs (binding), there are some differences in convergence times for all of the EA's examined. Relative to problem 1 (when the rough zone constraint is not binding), the convergence times for DE and PSO increase. For the ABCO algorithm, the mean convergence times are similar in both instances although the dispersion of the observation increases considerably. While for RCGA, the convergence time appears to decrease, relative to the nonbinding rough zone constraint case. Some degradation of the solution quality was apparent, particularly for the RCGA algorithm (See Appendix 13).

To reiterate, the results obtained in these and preceding experiments are relatively voluminous. These experimental results include not only the results described previously, but additional metrics of solution quality and dispersion. Further results from the release constraint experiments can be found in Appendix 13.

PSO Parameter Values

One of the more vexing aspects of working with EA's is the number of control parameters involved and identifying the values of these parameters which should be employed. While the literature provides a rich trove of advice on this topic, there is general agreement that parameter choice is problem specific.

The particle swarm optimization (PSO) algorithm has a relatively small set of parameters consisting of the cognitive weight parameter (c1) and the social weight parameter (c2). Both anecdotal and published empirical evidence suggest the solution success and performance can be quite sensitive to these parameters, depending on the specific problem it is applied to. The unit dispatch problem seems to be an example of such a problem.

Application of PSO to the unit dispatch problem, using the same parameter settings employed in Phase 1 of this project, produced unsatisfactory results. Following on this initial experience, a number of experiments were carried out including the systematic investigation of the social weight (c2) parameter. Figure 18 shown below illustrates the results of one series of replicated experiments designed to understand the effect of the c2 parameter on the performance of this algorithm.

As shown in Figure 18, the value of the c2 parameter used in Phase 1 of this research project (c2=1.30) and applied to the dynamic constrained economic dispatch problem, produces inferior results when employed on the unit dispatch problem. Based on this and other experiments carried out with the PSO algorithm, a c2 value of 1.50 was used for all Phase 2 experiments.

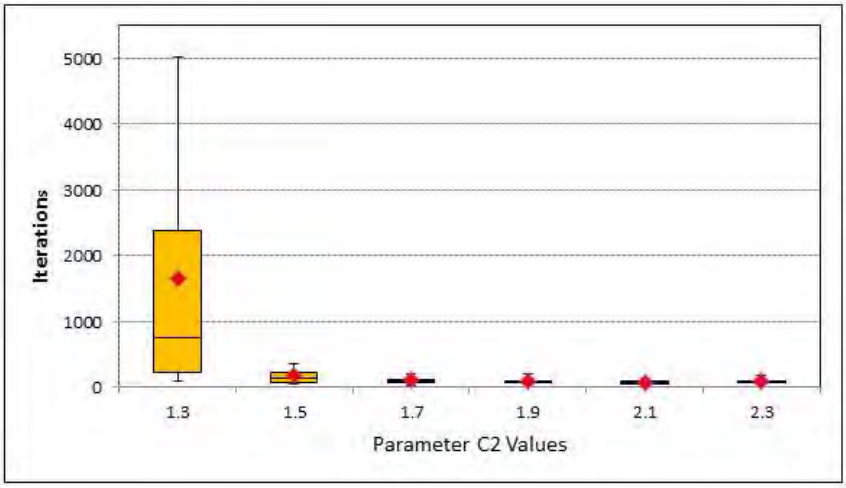

Figure 18.—PSO Parameter c2 Value Experiments

DE Mutation Strategies

There are a wide variety of available mutation strategies for the DE algorithm. The number of these seems to increase on a monthly basis, as do the related publications on this topic. While the peer reviewed literature describes many of these innovations, reports from applied studies seem to provide a better indicator of their efficacy.

For purposes of this research project, a subset of the known DE mutation strategies were implemented and investigated. The strategies investigated were the DE/Best/1, DE/Rand/1, DE/Best/2 and DE/Rand/2 strategies originally described in Price and Storn (1995), the trigonometric (Trigon) strategy (Fan and Lampinen 2003), the random scale factor (RandSF) and the time varying scale

factor (Detvsf) approaches of Das, Abraham and Konar (2008), and the self-adaptive (SelfAdapt) strategy of Brest et al (2006). These mutation strategies were selected based on their ease of implementation and a cursory assessment of how widely they were employed in applied studies.

Early experience with the application of DE to the unit dispatch problem indicated there were differences in performance between these mutation strategies. Following on this initial experience, a number of more systematic experiments were undertaken. Figure 19 shown below illustrates the results of replicated experiments designed to understand the effect of these mutation strategies on the performance of this algorithm.

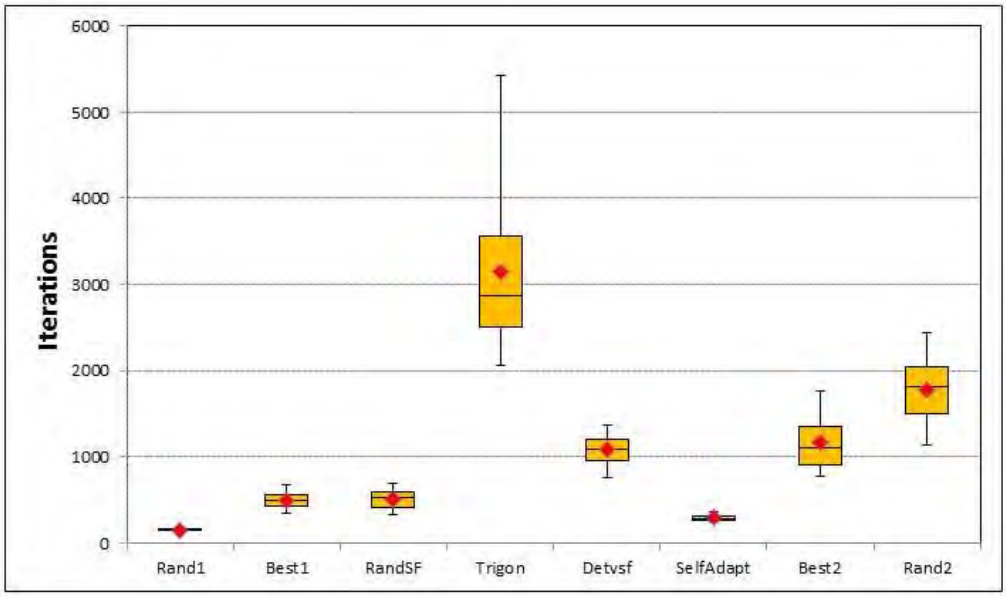

Figure 19.—Results of Mutation Strategy Experiments.

As shown in Figure 19, when applied to the unit dispatch problem the trigonometric (Trigon) mutation strategy is inferior to the others. The DE/Rand/1 and SelfAdapt strategies result in the best performance, followed by the DE/Best/1 the RandSF mutation strategies. The Detvsf , DE/Best/2 and DE/Rand/2 approaches appear to have similar performance characteristics. Based on this set of experiments with the DE algorithm, intuition about the approach mechanisms and a dose of professional judgment, the SelfAdapt mutation strategy (Brest et al 2006) was employed for all Phase 2 experiments.

Conclusions

Four promising evolutionary algorithms (EA's) were identified from the emerging heuristic optimization literature; the real coded genetic algorithm (RCGA), differential evolution (DE), particle swarm optimization (PSO) and the artificial bee colony optimization (ABCO) algorithm. These algorithms were applied to an important hydropower problem, the unit dispatch problem, which must be solved on a daily basis by power plant operators. A relatively extensive suite of replicated experiments was conducted to assess their performance characteristics. These experiments systematically explored the influence of initialization approaches, convergence criteria, dimensions of the problem and the effects of binding rough zone constraints. The aggregate experimental evidence indicates these algorithms can reliably solve the unit dispatch problem, within acceptable time-frames. Replicated experiments suggest different initialization approaches have no effect on solution times, for the problems examined. Stopping rule experiments indicate all of the rules which intelligently monitor convergence are superior to uninformed approaches. Replicated experiments indicate for DE, PSO and ABCO, convergence time increases for higher dimension problems. For the DE and PSO algorithms replicated experiments show convergence times increase when rough zone constraints are binding. For RCGA and ABCO, convergence times may decrease when rough zone constraints are binding. Many algorithm specific experiments were undertaken to inform this research effort. The results of a subset of these are also reported. Like the unit dispatch problem, many applied hydropower optimization problems are nonlinear, non-convex and discontinuous. These characteristics preclude the application of traditional calculus-based algorithms. In contrast, evolutionary algorithms are readily applied to this class of problem. They could provide near real-time solutions and guidance for everyday operational decisions at Reclamation's hydropower plants.

Future Directions

In aggregate, the replicated experimental results described here indicate the four selected evolutionary algorithms; PSO, RCGA, DE and ABCO are able to accurately and reliably solve the four unit dispatch test problems investigated. As described in this document, the dimensions of test problems (up to four units) are representative of 87.72 percent of the power plants owned by the Bureau of Reclamation. Some of the larger Reclamation power plants, including Glen Canyon (8-units), San Luis (8-units), Hoover (19-units) and Grand Coulee (24-units) are uniquely important components of the interconnected power system. For this reason, it may be advisable to assess the performance of these algorithms on problems with a greater number of generator units. The test problems examined here were limited to a subset of the operational constraints faced by operators in real-life. To more fully assess these algorithms, they should be employed on problems which reflect the full spectrum of constraints, including

consideration of the preferred dispatch order, must run units, spinning reserve, non-spinning reserve, regulation, start/stop costs and minimum up/down times.

Collaborators

Members of the collaborative research team played a pivotal role in the success of this research project. Their generous and invaluable technical contributions to this effort are gratefully acknowledged. The research team for this project was comprised of the following individuals.

Dr. Craig A. Bond, Full Economist
Rand Corporation, Inc.
1200 S. Hayes St, w7304
Arlington, VA 22202
(703) 413-1100 x5319
cbond@rand.org

Dr. Darrell G. Fontane, Professor
Department of Civil Engineering
Colorado State University
Ft. Collins, Colorado 80523
(970) 491-5247
fontane@engr.colostate.edu
Darrell.Fontane@ColoState.Edu

Dr. John B. Loomis, Professor
Department of Agricultural and Resource Economics
Colorado State University
Ft. Collins, Colorado 80523
(970) 491-2485
John.Loomis@colostate.edu

Mr. Thomas D. Veselka, Senior Power Engineer
DIS Division
Argonne National Laboratory
Argonne, Illinois 60439
(630) 252-3449
tdveselka@anl.gov

Literature Cited

Abookazemi, Kaveh, Mohd W. Mustafa and Hussein Ahmed. "Structured Genetic Algorithm Technique for Unit Commitment Problem." *International Journal of Recent Trends in Engineering* Vol. 1 No. 3 (May 2009):135-139.

Abraham, Ajith, Ravi Kumar Jatoth and A. Rajasekhar. "Hybrid Differential Artificial Bee Colony Algorithm." *Journal of Computational and Theoretical Nanoscience* Vol 9 No. 2 (February 2012):1-9.

Akay, Bahriye and Dervis Karaboga. "Artificial Bee Colony Algorithm for Large-Scale Problems and Engineering Design Optimization." *Journal of Intelligent Manufacturing*, In Press, doi:10.1007/s10845-010-0393-4

Ali, Musrrat, Millie Pant and Ajith Abraham. "A Hybrid Ant Colony Differential Evolution and its Application to Water Resource Problems." proceedings of the Nature & Biologically Inspired Computing NaBIC 2009 World Congress. Coimbatore, India. 9-11 December. 2009 pages 1133 – 1138.

Alzaqebah, Malek and Salwani Abdullah. "Comparison on the Selection Strategies in the Artificial Bee Colony Algorithm for Examination Timetabling Problems." *International Journal of Soft Computing and Engineering (IJSCE)* Vol 1 No. 5 (September 2011): 158-163.

American Society of Mechanical Engineers. *Hydraulic Turbines and Pump-Turbines Performance Test Codes.* ASME PTC 18-2011. American Society of Mechanical Engineers, Three Park Avenue . New York, NY. 2011.

Ao, Youyun and Hongqin Chi. "Dynamic Differential Evolution for Constrained Real-Parameter Optimization." *Journal of Advances in Information Technology* Vol.1 No. 1. (February 2010):43-51.

Bacanin, Nebojsa, Milan Tuba and Ivona Brajevic. "Performance of Object-Oriented Software System for Improved Artificial Bee Colony Optimization." *International Journal of Mathematics and Computers in Simulation* Vol 5 No. 2 (2011):154-162.

Baloi, Cristian A., John A. Belward, and Michael Bulmer, M. "Genetic Unit Commitment Model In A Deregulated Power Energy Environment." In, Proceedings of International Congress On Modelling And Simulation (MODSIM 2003). International Congress on Modelling & Simulation (MODSIM 2003), Townsville, Australia, (1078-1083). 14-17 July 2003.

Banacos, Peter C. "Box and Whisker Plots for Local Climate Datasets: Interpretation and Creation Using Excel 2007/2010." Eastern Regional Technical Attachment No. 2011-01. National Oceanographic and

Atmospheric Administration (NOAA)/National Weather Forecast Service (NWS). Weather Forecast Office. South Burlington, Vermont. January 2011. 20 pages. obtained from: www.erh.noaa.gov/er/hq/ssd/erps/ta/ta2011-01.pdf. Accessed on 2/20/2012.

Banks, Alec, Jonathan Vincent and Chukwudi Anyakoha. "A Review of Particle Swarm Optimization. Part I: Background and Development. *Natural Computing* Vol 6 No. 4 (December 2007): 467-484.

Banks, Alec, Jonathan Vincent and Chukwudi Anyakoha. "A Review of Particle Swarm Optimization. Part II: Hydridization, Combinatorial, Multicriteria and Constrained Optimization, and Indicative Applications." *Natural Computing* Vol 7 No. 1 (March 2008): 467-484.

Blum, Christian and Xiaodong Li. "Swarm Intelligence in Optimization." Chapter 2 in, *Swarm Intelligence—Introduction and Applications.* Christian Blum and Daniel Merkle (Eds). Natural Computing Series. Leiden Center for Natural Computing. Springer-Verlag, Berlin. 2008.

Blum, Christian. "Ant Colony Optimization: Introduction and Recent Trends." *Physics of Life Reviews* Vol 2 No. 4 (December 2005):353-373.

Blackwell, Tim Jurgen Branke and Xiaodong Li. "Particle Swarms for Dynamic Optimization Problems." In, *Swarm Intelligence—Introduction and Applications.* Springer Natural Computing Series. Leiden Center for Natural Computing. Christian Blum and Daniel Merkle, Editors. Springer-Verlang: Berlin, Germany. 2008.

Bouzeboudja, Hamid, Abdelkader Chaker, Ahmed Allall, and Bukhta Naama. "Economic Dispatch Solution Using A Real-Coded Genetic Algorithm." *Acta Electrotechnica et Informatica* Vol. 5 No. 4 (2005):1-5.

Boyang Li, Yew-Soon Ong, Minh Nghia Le, Chi Keong Goh: Memetic Gradient Search. IEEE Congress on Evolutionary Computation 2008: 2894-2901.

Boyd, Stephen and Lieven Vandenberghe. *Convex Optimization*. New York City, New York: Cambridge University Press. 2004. (revised 2006). 730 pages.

Bratton, Don and Tim Blackwell. "A Simplified Recombinant PSO." In, *Swarm Intelligence—Introduction and Applications.* Springer Natural Computing Series. Leiden Center for Natural Computing. Christian Blum and Daniel Merkle, Editors. Springer-Verlang: Berlin, Germany. 2008.

Brajevic, Ivona and Milan Tuba. "An Upgraded Artificial Bee Colony (ABC) Algorithm for Constrained Optimization Problems." *Journal of Intelligent Manufacturing* Vol 24 No. 4 (August 2013):729-740.

Brest, Jamez, Viljem Zumer and Mirjam Sepesy Maucec. "Self-Adaptive Differential Evolution Algorithm in Constrained Real-Parameter Optimization." Pages 919-926 in, Proceedings of the 2006 IEEE Congress on Evolutionary Computation. Vancouver, British Columbia, Canada. July 16-21, 2006.

Bucknall, Julian. "Ant Colony Optimizations." *The Delphi Magazine* Issue 136 (December 2006):17-22.

Bucknall, Julian. "Round & Round—How Random are Your Numbers?" *The Delphi Magazine* Issue 33 (May 1988):18-25.

Caldwell, Chris. "The Prime Pages—Prime Number Research, Records and Resources." University of Tennessee. http://primes.utm.edu . Last accessed on 12/17/2009.

Carlisle, Anthony and Gerry Dozier. "An Off-The-Shelf PSO." Proceedings of the Workshop on Particle Swarm Optimization. Indianapolis, IN. 2001.

Chakraborty, Uday K (Editor), *Advances in Differential Evolution.* Studies in Computational Intelligence, Volume 143. Springer-Verlang: Belin, Germany. 2008.

Chand, Pramesh and Ly Fie Sugianto. "Horizon-Scan: In Search For Consistent Outcomes." *International Journal of Computational Intelligence Research* Vol.4, No.3 (2008), pp. 256–272.

Chandrum, K., N. Subrahmanyam and M. Sydulu. "Brent Method for Dynamic Economic Dispatch with Transmission Losses." *Iranian Journal of Electrical and Computer Engineering* Vol. 8 No. 1 (Winter-Spring 2009):16-22,

Coelho, Leandro do Santos and Viviana C. Mariani. "Improved Differential Algorithms for Handling Economic Dispatch Optimization with Generator Constraints." *Energy Conversion and Management.* Vol. 48 No. 5 (May 2007):1631-1639.

Coello Coello, Carlos A. "Theoretical and Numerical Constraint-Handling Techniques Used With Evolutionary Algorithms: A Survey of the State Of the Art." *Computer Methods in Applied Mechanics and Engineering* 191 No. 11-12 (January 2002): 1245-1287.

Computer Dictionary Online. Web based dictionary of computer terms. Located at: www.computer-dictionary-online.org Last accessed on 28 September 2010.

Clerc, Maurice. "Initializations for Particle Swarm Optimization." Unpublished manuscript. 24 December 2008. Available from: http://clerc.maurice.free.fr/pso/. Last accessed on 12/31/2009.

Clerc, Maurice. "Confinements and Biases in Particle Swarm Optimization." Unpublished manuscript. 12 March 2006. Available from: http://clerc.maurice.free.fr/pso/ . Last accessed on 01/04/2010.

Clerc, Maurice. *Particle Swarm Optimization*. London, England: ISTE Publishing Company. 2006.

Clerc, Maurice and James Kennedy. "The Particle Swarm—Explosion, Stability and Convergence in a Multidimensional Complex Space." *IEEE Transactions on Evolutionary Computation* Vol 6 No. 1 (February 2002): 58-73.

Cutello, Vincenzo and Giuseppe Nicosia "An Immunological Approach to Combinatorial Optimization Problems" in, *Lecture Notes in Computer Science*, Vol. 2527. Proceedings of the 8th Ibero-American Conference on AI: Advances in Artificial Intelligence. 2002 pp. 361–370.

Curtis, Michael, Jonas Parker and Parker Scoggins. "Optimizing Operations of Multi-Unit Powerhouses: A New Method." *Hydro Review* Vol 31 No. 5 (July 2012):1-8

Damousis, Ionnis G., Anastasios G. Bakirzis and Petros S. Dokopoulos. "Network-Constrained Economic Dispatch Using Real-Coded Genetic Algorithm." *IEEE Transactions on Power Systems* Vol. 18 No. 1 (February 2003): 198-205.

Das, Swagatam, Ajith Abraham and Amit Konar. "Particle Swarm Optimization and Differential Evolution Algorithms: Technical Analysis, Applications and Hybridization Perspectives." *Advances of Computational Intelligence in Industrial Systems* Vol. 116. Ying Liu et al. (Eds.), Studies in Computational Intelligence, Springer Verlag, Germany, 2008. pp. 1–38.

Davidovic, Tatjana, Milica Selmic, Dusan Teodorovic, Dusan Ramljak: "Bee Colony Optimization For Scheduling Independent Tasks To Identical Processors.". *Journal of Heuristics* Vol 18 No. 4 (August 2012):549-569.

Deb, Kalyanmoy. "An Efficient Constraint Handling Method for Genetic Algorithms." *Computer Methods in Applied Mechanics and Engineering* Vol. 186 No. (2-4) (June 2000): 311-338.

Deep, Kusum and Manoj Thakur. "A New Crossover Operator for Real Coded Genetic Algorithms." *Applied Mathematics and Computation* Vol. 188 No. 1 (May 2007):895-911.

De Jong, Kenneth A. "Analysis of the Behavior of a Class of Genetic Adaptive Systems." Unpublished Ph.D. Dissertation. Computer and Information Sciences. University of Michigan, Ann Arbor. 1975.

Dorigo, Marco and Thomas Stutzle. *Ant Colony Optimization*: MIT Press, Inc. July 2004. 319 pages.

Eberhart, Russell C. and James Kennedy. "A New Optimizer Using Particle Swarm Theory." Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, October 1995; 39-43.

Edwards, Brian K. *The Economics of Hydroelectric Power*. New Horizons in Environmental Economics. Northampton, Massachusetts: Edward Elgar Publishing Inc. 2003.

Edwards, Brian K., Silvio J. Flaim, and Richard E. Howitt. "Optimal Provision of Hydroelectric Power Under Environmental and Regulatory Constraints." *Land Economics*. Vol 75 No. 2 (May 1999):267-283.

Edwards, Brian K., Richard E. Howitt, and Silvio J. Flaim. "Fuel, Crop, and Water Substitution in Irrigated Agriculture." *Resource and Energy Economics* 18 No. 3 (October 1996):311-331.

Engelbrecht, Andries P. *Fundamentals of Computational Swarm Intelligence*. Hoboken, New Jersey: John Wiley & Sons, Ltd. 2005.

Farmani, Raziyeh and Jonathan A. Wright. "Self-Adaptive Fitness Formulation for Constrained Optimization." *IEEE Transactions on Evolutionary Computation* Vol. 7 No. 5 (October 2003):445-455.

Feoktistov, Vitaliy. *Differential Evolution—In Search of Solutions* Volume 5 in Optimization and its Applications Series. Panos M. Pardalos, Managing Editor. Springer, New York, NY. 2006.

Forsund, Finn R. *Hydropower Economics* International Series in Operations Research and Management Science. Frederic S. Hillier, Series Editor. New York, NY. Springer. 2010.

Finardi, Erlon C. and Edson Luiz da Silva. "Solving the Hydro Unit Commitment Problem via Dual Decomposition and Sequential Quadratic Programming." *IEEE Transactions On Power Systems*, Vol. 21 No. 2 (May 2006):835-844.

Fylstra, Daniel, Leon Lasdon, John Watson and Allen Warren. "Design and Use of the Microsoft Excel Solver." *Interfaces* 28 No. 5 (September-October) 1998:29-55.

General Electric Energy Corporation. *Western Wind And Solar Integration Study* Prepared for the National Renewable Energy Laboragory. GE Energy. Schenectady, NY. May 2010. 536 pages.

General Electric Energy Corporation. MAPS™ Multi-Area Production Simulation Model Product Description. GE Energy. Schenectady, NY. March 2008. 33 pages.

Goldberg, David E. *Genetic Algorithms in Search Optimization, and Machine Learning*. Reading, Massachusetts: Addison-Wesley Professional Inc. 1989 (reissued). 432 pages.

Gong, Wenyin, Alvaro Fialho and Zhihua Cai. "Adaptive Strategy Selection in Differential Evolution." Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) 2010. July 7-11, 2010. Portland, Oregon. ACM Press, Inc. 2010.

Haddad, Omid. Bozorg. Abbas Afshar and Miguel A. Marino. "Optimization of Non-Convex Water Resource Problems by Honey-Bee Mating Optimization (HBMO) Algorithm." *Engineering Computations* Vol. 26 No. 3 (2009):267-280.

Haddad, Omid Bozorg, Abbas Afshar and Miguel A. Marino. "Honey-Bee Mating Optimization (HBMO) Algorithm: A New Heuristic Approach for Water Resources Optimization." *Water Resources Management* Vol 20 No. 5 (October 2006):661-680.

Harpman, David A. *Advanced Algorithms for Hydropower Dispatch (Phase 1)* Technical Report S&T-2011-486. U.S. Bureau of Reclamation, Technical Service Center, Denver, Colorado. March 2012. 136 pages.

Harpman, David A. "Assessing the Short-Run Economic Cost of Environmental Constraints on Hydropower Operations at Glen Canyon Dam." *Land Economics* 75 No. 3 (August 1999):390-401.

Haupt, Randy L. and Sue Ellen Haupt. *Practical Genetic Algorithms*. 2nd Edition. John Wiley and Son, Inc, New York, N.Y. 2004. 192 pages.

Helwig, Sabine and Rolf Wanka. "Particle Swarm Optimization in High-Dimensional Bounded Search Spaces." Proceedings of the 2007 IEEE

Swarm Intelligence Symposium. 1-5 April 2007 Honolulu, HI, Pages 198-205.

Hemamalini, S. and Sishaj P. Simon. "Dynamic Economic Dispatch Using Artificial Bee Colony Algorithm for Units with Valve-Point Effect." *European Transactions on Electrical Power* Vo. 21 No. 1 (January 2011):70-81.

Holland, John H. *Adaption in Natural and Artificial Systems*. Ann Arbor: University of Michigan Press. 1975.

Hu, Xiaohui and Russell Eberhart. "Solving Constrained Nonlinear Optimization Problems with Particle Swarm Optimization." Pages 203-206 in, Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2002), Orlando, USA. 2002.

Huang, Hua-Juan and Yong-Quan Zhou. "Hybrid Artificial Fish Swarm Algorithm For Global Optimization Problems" *Journal of Computer Applications*. Vol. 28, no. 12 (December 2008): 3062-3064.

Hulse, David O. Manager, Mechanical Equipment Group 86-68420. Reclamation Technical Service Center. Denver, Colorado. Personal communication with David A. Harpman, Natural Resource Economist. Economics, Planning and Technical Communications Group 86-68270. September 30, 2013.

International Electrotechnical Commission. *International Standard for Hydraulic Turbines, Storage Pumps and Pump-Turbines – Model Acceptance Tests* CEI/IEC 60193:1999 2nd Edition. International Electrotechnical Commission 3, rue de Varembé Geneva, Switzerland. 1999.

Jeyakumar, D.N., T. Jayabarathi and T. Raghunathan. "Particle Swarm Optimization for Various Types of Economic Dispatch Problems." *International Journal of Electrical Power and Energy Systems* Vol. 28 No. 1 (January 2006):36-42.

Judd, Kenneth L. *Numerical Methods in Economics*. Second Printing. Cambridge, Massachusetts: MIT Press. 1999.

Karaboga, Dervis, Beyza Gorkemli, Celal Ozturk and Nurhan Karaboga. "A Comprehensive Survey: Artificial Bee Colony (ABC) Algorithm and Applications" *Artificial Intelligence Review* published online DOI: 10.1007/s10462-012-9328-0 March 2012

Karaboga, Dervis and Bahriye Akay, "A Modified Artificial Bee Colony (ABC) Algorithm for Constrained Optimization Problems." *Applied Soft Computing* Vol 11 No. 3 (April 2011):3021-3031.

Karaboga, Dervis and Bahriye Basturk. “A Comparative Study of Artificial Bee Colony Algorithm.” *Applied Mathematics and Computation* Vol 214 No. 1 (August 2009):108-112.

Karaboga, Dervis and Bahriye Akay “ ASurvey: Algorithms Simulating Bee Swarm Intelligence.” *Artificial Intelligence Review* Vol 31, Issue 1-4 (June 2009): 61-85

Karaboga, Dervis. and Bahriye. Akay. "Artificial Bee Colony (ABC), Harmony Search And Bees Algorithms On Numerical Optimization." *Proceedings of Innovative Production Machines and Systems* Virtual Conference, IPROMS. 2009. July 6 to July 17, 2009. Cardiff, United Kingdom.

Karaboga, Dervis and Bahriye Akay “ A Comparative Study of Artificial Bee Colony Algorithm.” *Applied Mathematics and Computation* Vol 214, No. 1 (August 2009): 108-132

Karaboga, Dervis and Bahriye Basturk. “A Powerful and Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony (ABC) Algorithm.” *Journal of Global Optimization* 39 No. 3 (November 2007):459-471.

Karaboga, Dervis and Bahriye Basturk. “Artificial Bee Colony (ABC) Optimization Algorithm for Solving Constrained Optimization Problems.” *Advances in Soft Computing: Foundations of Fuzzy Logic and Soft Computing.* Lecture Notes in Computer Science Vol: 4529, pp: 789-798, Springer- Verlag, 2007.

Karaboga, Dervis. “An Idea Based On Honey Bee Swarm For Numerical Optimization.” Technical Report-Tr06, Computer Engineering Department, Erciyes University. Kayseri, Turkey. October 2005.

Kaur, Arvinder * and Shivangi Goyal. “A Survey on the Applications of Bee Colony Optimization Techniques.” *International Journal on Computer Science and Engineering* Vol 3 No. 8 (August 2011):3037-3045.

Kennedy, James and Rui Mendes. “Population Structure and Particle Swarm Performance.” Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress. Honolulu, HI , USA. 12 May 2002 - 17 May 2002. pages 1671 – 1676.

Kennedy, James and Russell C. Eberhart. *Swarm Intelligence*. San Francisco, CA: Morgan Kaufmann Academic Press, 2001.

Kim, Dong Hwa, Ajith Abraham and Jae Hoon Cho. "A Hybrid Genetic Algorithm and Bacterial Foraging Approach for Global Optimization." *Information Sciences* Vol 177 No. 18 (September 2007): 3918–3937.

Kirkpatrick, Scott, Charles D. Gelatt, M. P. Vecchi. "Optimization by Simulated Annealing" *Science*, New Series, Vol. 220 No. 4598 (May 1983): 671-680.

Klimasauskas, Casimir C. "Not Knowing Your Random Number Generator Could be Costly: Random Generators-- Why They are Important." *Personal Computer Artificial Intelligence* Vol 16 No. 3 (May/June 2002):52-59.

Knuth, Donald E. *The Art of Computer Programming: Seminumerical Algorithms* Vol 2. 3rd Edition. Massachusetts: Addison-Wesley Press, Inc. 2002.

Kumar, Awadhesh. "Dynamic Economic Dispatch Using Particle Swarm Optimization." Unpublished Masters Thesis. Electrical and Instrumentation Engineering Department, Thapar University, Patiala. June 2009.

Lee, Kwang Y. and Jong-Bac Park. "Application of Particle Swarm Optimization to Economic Dispatch Problem: Advantages and Disadvantages." in, proceedings of the Power Systems Conference and Exposition, 2006. Atlanta, GA. Oct. 29 - Nov. 1 2006. pages 188 – 192.

Li, Boyang, Yew-Soon Ong, Minh Nghia Lee and Chi Keong Goh. "Memetic Gradient Search." *Evolutionary Computation 2008*. Proceedings of the 2008 IEEE World Congress on Computational Intelligence. Hong Kong, China. June 1-6, 2008. pages 2894 – 2901.

LINDO Systems. "Electrical Generation Unit Commitment Planning." Application Survey Paper. LINDO Systems, Inc. Chicago, Illinois. June 2003.

Liu, Hui, Zixing Cai and Yong Wang. "Hybridizing Particle Swarm Optimization with Differential Evolution for Constrained Numerical and Engineering Optimization." *Applied Soft Computing* Vol 10 No. 2 (March 2010):629-640.

Lucic, Panta and Dusan Teodorovic. "Computing with Bees: Attacking Complex Transportation Engineering Problems." *International Journal of Artificial Intelligence Tools* Vol 12 No 3 (September 2003):375-394.

Ma, Xin and Young Liu. "Economic Dispatch Considering Ancillary Service Based on Revised Particle Swarm Optimization Algorithm." in, Proceedings of the 6th International Conference on Advanced Intelligent Computing Theories And Applications: Intelligent Computing Changsha, China, August 18-21, 2010. Pages 175-184.

Matsumoto, Makoto and Takuji Nishimura. "Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudorandom Number Generator." *Association for Computing Machinery (ACM) Transactions on Modeling and Computer Simulations* Vol 8 No. 1 (January 1998):3-30.

Mezura-Montes, Efren, and Omar Cetina-Domınguez. "Empirical Analysis of a Modified Artificial Bee Colony for Constrained Numerical Optimization." *Applied Mathematics and Computation* Vol 22 No. 15 (July 2012): 10943–10973.

Mezura-Montes, Efren, and Omar Cetina-Domınguez. "Exploring Promising Regions Of The Search Space With The Scout Bee In The Artificial Bee Colony For Constrained Optimization." In *Proceedings of the Artificial Neural Networks in Enginnering Conference* (ANNIE'2009), vol. 19, pp. 253-260. 2009.

Mezura-Montes, Efren and Jorge Isacc Flores-Mendoza. "Improved Particle Swarm Optimization in Constrained Numerical Search Spaces." in Raymond Chiong (Editor), *Nature-Inspired Algorithms for Optimisation*, pages: 299-332, Springer-Verlag, Studies in Computational Intelligence Series Vol. 193, 2009.

Mezura-Montes, Efren and Blanca Cecilia Lopez-Ramirez. "Comparing Bio-Inspired Algorithms in Constrained Optimization Problems." Pages 662-669 in, Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2007, 25-28 September 2007, Singapore. 2007

Mezura-Montes, Efren, Jesus Velazquez-Reyes and Carlos A. Coello Coello. "Modified Differential Evolution for Contrained Optimization" Pages 25-32 in, Proceedings of the 2006 IEEE Congress on Evolutionary Computation. Vancouver, British Columbia, Canada. July 16-21, 2006.

Michalewicz, Zbigniew. *Genetic Algorithms + Data Structures = Evolution Programs.* 3rd Ed. Springer. 1996. 387 pages.

Michalewicz, Zbigniew and David B. Fogel. *How to Solve it: Modern Heuristics.* 2nd Ed. Springer. 2010. 554 pages.

Mishra, S.K. "Global Optimization by Differential Evolution and Particle Swarm Methods Evaluation on Some Benchmark Functions." MPRA Paper No. 1005, posted 07 November 2007. http://mpra.ub.uni-muenchen.de/1005.

Miranda, Mario J. and Paul L. Fackler. *Applied Computational Economics and Finance.* Cambridge, Massachusetts: MIT Press. 2006.

Miranda, Vladimiro. "Operational Planning: Unit Commitment and Economic Dispatch." Chapter 12 in, *Tutorial on Modern Heuristic Optimization Techniques with Applications to Power Systems.* IEEE Power Engineering Society 02TP160. Edited by Kwang Y. Lee and Mohamed A. El-Sharkawi. Institute of Electrical and Electronics Engineers (IEEE), Power Engineering Society. pages 130-137.

Monson, Christopher K. and Kevin D. Seppi. "Linear Equality Constraints and Homomorphous Mappings in PSO." in, Proceedings of the 2005 IEEE World Congress on Evolutionary Computation, Vol. 1 pages 73-80. Edinburgh, Scotland. September 5, 2005.

Moussa, Abd-El Moneim, Mahmoud A. El-Gammal, Amr Y. Abou-Ghazala and Amani I. Attia. "A Novel Approach for Unit Commitment Problem via an Effective Modified Particle Swarm Optimization Technique." *European Journal of Scientific Research* Vol. 48 No. 4 (January 2011):546-558. http://conference.iproms.org/forums/iproms_2006/optimisation_techniques Last accessed on 12/29/2009.

Nelder, John A. and Roger Mead. "A Simplex Method for Function Minimization" *Computer Journal* 7 No. 4 (January 1965):308-313.

Neri, Ferrante and Ville Tirronen. "Recent Advances in Differential Evolution: A Survey and Experimental Analysis." *Artificial Intelligence Review* 33 Nos. 1-2 (February 2010): 61-106.

Nguyen, Q. H., Yew-Soon Ong, Natalio Krasnogor. "A Study on the Design Issues of Memetic Algorithm." IEEE Congress on Evolutionary Computation 2007: 2390-2397

Noman, Nasimul and Htoshi Iba. "Accelerating Differential Evolution Using Adaptive Local Search." *IEEE Transactions on Evolutionary Computation* Vol. 12 No. 1 (February 2008):107-125.

North American Electricity Reliability Council (NERC). *Balancing and Frequency Control.* A Technical Document Prepared by the NERC Resources Subcommittee. January 26, 2011. NERC. Princeton, N.J. Obtained from: www.nerc.com/docs Accessed on 25 March 2013.

Omran, Mahamed G.H. "Using Opposition-based Learning with Particle Swarm Optimization and Barebones Differential Evolution." Chapter 23 in, *Particle Swarm Optimization*. Edited by Aleksandar Lazinica. Vienna, Austria: INTECH- Education and Publishing. January 2009. 476 pages.

Omran, Mahamed G.H., Andries P. Engelbrecht and Ayed Salman. "Bare Bones Differential Evolution." *European Journal of Operations Research* Vol 196 No. 1 (July 2009):128-139.

Paquet, Ulrich and Andries P. Engelbrecht. "Particle Swarms for Linearly Constrained Optimization." *Fundamenta Informaticae* Vol. 76 No. 1-2 (March 2007):147-170.

Paquet, Ulrich and Andries P. Engelbrecht. "A New Particle Swarm Optimiser for Linearly Constrained Optimization." in, proceedings of the 2003 IEEE World Congress on Evolutionary Computation Vol. 1 pages 227-233. Canberra, Australia. December 8-13, 2003.

Pant, Millie, Radha Thangaraj, Ved Pal Singh and Alith Abraham. "Particle Swarm Optimization Using Sobol Mutation." Pages 367 to 372 in, The Proceedings of the 2008 First International Conference on Emerging Trends in Engineering and Technology. Nagpur, India. 16-18 July 2008.

Pant, Millie, Radha Thangaraj and Alith Abraham. "Low Discrepancy Initialized Particle Swarm Optimization for Solving Constrained Optimization Problems." *Fundamenta Informaticae* 95 No. 4 (December 2009):1-21.

Pant, Millie, Ved Pal Singh and Alith Abraham. "Differential Evolution using Quadratic Interpolation for Initializing the Population." Pages 375 to 380 in, Proceedings of the 2009 Advance Computing Conference, IACC 2009. IEEE International. Delhi, India. 6-7 March 2009.

Park, Stephen K. and Keith W. Miller, "Random Number Generators: Good Ones Are Hard to Find." *Communications of the Association for Computing Machinery (ACM),* Vol. 31 No. 10 (October 1988): 1192-1201.

Parsopoulos, Konstantinos E, and Michel N. Vrahatis. "Initializing the Particle Swarm Optimizer Using the Nonlinear Simplex Method" In, *Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation*. The Artificial Intelligence Series. Edited by A. Grmela and N.E. Mastorakis. World Scientific and Engineering Academy and Society (WSEAS) Press: Interlaken, Switzerland. 2002.

Pedersen, Magnus Erik Hvass. "Tuning & Simplifying Heuristical Optimization." Unpublished Ph.D. Thesis. School of Engineering Sciences, Computational Engineering and Design Group. University of Southampton. England. January 2010.

Peltokangas Riikka and Aki Sorsa. "Real-coded Genetic Algorithms and Nonlinear Parameter Identification." Report A No. 48. Control Engineering Laboratory, University of Oulu. April 2008.

Pham, D.T., A. Ghanbarzadeh, E. Koc., S. Otri, S. Rahim and M. Zaidi. "The Bees Algorithm—A Novel Tool for Complex Optimization Problems." Innovative Production Machines and Systems 2006 Virtual Conference. July 2006. Available from: http://conference.iproms.org/forums/iproms_2006/optimisation_techniques. Last accessed on 12/29/2009.

Pomeroy, Paul. "An Introduction to Particle Swarm Optimization." March 2003. Online article accessed at: www.adaptiveview.com/articles.

Potter, Walter D., Eric Drucker, Pete Bettinger, Frederick Maier, Max Martin, D. Luper, M. Watkinson, G. Handy, C. Hayes. "Diagnosis, Configuration, Planning, and Pathfinding: Experiments in Nature-Inspired Optimization." *in, Natural Intelligence for Scheduling, Planning and Packing Problems.* Studies in Computational Intelligence. Vol. 250. Berlin, Germany: Springer Verlag. 2009. pages 267-294.

Pourakbari-Kasmaei, Mahdi, Masoud Rashidi-Nejad and Amir Abdollahi. "A Novel Unit Commitment Technique Considering Prohibited Operating Zones." *Journal of Applied Sciences* Vol. 9 No. 16 (2009):2962-2968.

Press, William H., Brian P. Flannery, Saul A. Teukolsky and William T. Vetterling. *Numerical Recipes in Pascal—The Art of Scientific Computing.* New York: Cambridge University Press. 1989.

Price, Kenneth V. and Rainer M. Storn. "Differential Evolution" *Dr. Dobb's Journal* Issue 264 (April 1997):18-24 and 78.

Price, Kenneth V., Rainer M. Storn and Jouni A. Lampinen. *Differential Evolution - A Practical Approach to Global Optimization.* Springer-Verlang: Belin, Germany. 2005. 538 pages.

Puri, Vinod. "Unit Commitment Using Particle Swarm Optimization." Unpublished Thesis. Electrical and Instrumentation Engineering Department, Thapar University. Patiala, India. July 2009.

Rabiei, Abbas, Alireza Soroudi and Behnam Mohammadi. "Imperialist Competition Algorithm for Solving Non-convex Dynamic Economic Power Dispatch." 11-E-PSS-1921. Proceedings of the 26th International Power System Conference. 31 October to 2 November 2011. Tehran, Iran. 2011. Pages 1-8.

Rahnamayan, Shahryar and G. Gary Wang. "Solving Large Scale Optimization Problems by Opposition-Based Differential Evolution (ODE). "*WSEAS Transactions on Computers* 10 Vol. 7 (October 2008):1792-1804.

Rahnamayan, Shahryar, Hamid R. Tizhoosh and Magdy M.A. Salama. "Opposition-Based Differential Evolution." Chapter 6 in, *Advances in Differential Evolution.* Studies in Computational Intelligence, Volume 143. Edited by Uday K. Chakraborty. Springer-Verlang: Belin, Germany. 2008.

Rajan, C.Christober Asir. "Genetic Algorithm Based Simulated Annealing Method for Solving Unit Commitment Problem in Utility System." *International J. of Recent Trends in Engineering and Technology*, Vol. 3, No. 4 (May 2010):96-100.

Rajkumar, N. Timo Vekara and Jarmo T. Alander. "A Review of Genetic Algorithms in Power Engineering." in, *AI and Machine Consciousness—Proceedings of the 13th Finish Artificial Intelligence Conference.* Tapani Raiko, Penti Haikonen and Jaakko Vayrynen, editors. Esppo, Finland. August 20-22, 2008. pages 15-32.

Reeves, Collin R. "Genetic Algorithms." Chapter 3 in, *Handbook of Metaheuristics* 2nd Edition. edited by Michel Gendreau and Jean-Yves Potvin. Springer Business and Science: New York City, NY. 2010. 650 pages.

Rau, Narayan S. *Optimization Principles—Practical Applications to the Operation and Markets of the Electric Power Industry.* IEEE Press Series on Power Engineering. P.M. Anderson, Series Editor. New York, New York: John Wiley and Sons. 2003. 339 pages.

Richards, Mark and Dan Ventura. "Choosing a Starting Configuration for Particle Swarm Optimization." pages 2309–2312 in, *Proceedings of the Joint Conference on Neural Networks.* July 2004.

Robertson, Grant. "How Powerful was the Apollo 11 Computer?" *Download Squad*. Weblogs, Inc. RSS Feed. July 20, 2009 at 8:30 pm.

Saxena, D., Sri N. Singh and K. S. Verma. "Application of Computational Intelligence in Emerging Power Systems." *International Journal of Engineering, Science and Technology* Vol. 2 No. 3 (May 2010):1-7.

Shah-Hosseini, Hamedi, "The Intelligent Water Drops Algorithm: A Nature-Inspired Swarm-Based Optimization Algorithm." *International Journal of Bio-Inspired Computation,* Vol. 1, Nos. 1 and 2 (January 2009):71-79.

Sharma, Tarun Kumar, Millie Pant and V.P. Singh. "Adaptive Bee Colony in an Artificial Bee Colony for Solving Engineering Design Problems." *Advances in Computational Mathematics and its Applications (ACMA)* Vol 1 No. 4 (2012):213-221.-

Sharma, Nidhi, Kalpana Jain and Manjaree Pandit. "Artificial Bee Colony Optimization for Static Economic Dispatch." *International Journal of Advanced Computational Methods and Applications* Vol 1 No. 1 (2011):16-25.

Simon, Sishaj P., Narayana P. Padhy and R. S. Anand. "An Ant Colony System Approach for the Unit Commitment Problem." *International Journal of Electrical Power Energy Systems* Vol. 28 No. 5 (June 2006): 315-323

Simopoulos, Dimitris N. Stavroula D. Kavatza and Costas D. Voumas. "An Enhanced Peak Shaving Method for Short Term Hydrothermal Scheduling." *Energy Conversion and Management* Vol. 40 No. 1 (November 2007): 2018-3024.

Siu, Thomas K., Garth A. Nash and Ziad K. Shawwash. "A Practical Hydro, Dynamic Unit Commitment and Loading Model." *IEEE Transactions on Power Systems* Vol. 16 No. 2 (May 2001): 301-306.

Socha, Krzysztof and Marco Dorigo. "Ant Colony Optimization for Continuous Domains." *European Journal of Operational Research* Vo. 185 No. 2 (March 2008): 1155-1173.

Sriyanyong, Pichet, Yan-Hong. Song and Paul J. Turner. "Particle Swarm Optimization for Operational Planning: Unit Commitment and Economic Dispatch." *Studies in Computational Intelligence* Vol. 49 (2007):313-347.

Stanarevic, Nadezda, Milan Tuba and Nebojsa Bacanin. "Modified Artificial Bee Colony Algorithm for Constrained Problems Optimization." *International Journal of Mathematical Models and Methods in Applied Sciences* Vol. 5 No. 3 (2011):644-651.

Staschus, Konstantin, Andrew M. Bell, and Eileen Cashman. "Usable Hydro Capacity, Electric Utility Production Simulations, and Reliability Calculations." Institute of Electrical and Electronics Engineers (IEEE), *Transactions on Power Systems* Vol 5 No. 2 (May 1990):531-538.

Steves, Toby L. Electrical Engineer, Hydropower Technical Services Group 86-68450, U.S. Bureau of Reclamation. Technical Service Center, Denver, Colorado. Email message to David A. Harpman on April 5, 2012

Storn, Rainer M. and Kenneth V. Price. *Differential Evolution—A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces.* Technical Report TR-95-12, International Computer Science Institute. March 1995. Available from: http://www.icsi.berkeley.edu .

Storn, Rainer M. and Kenneth V. Price. "Differential Evolution—A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces." *Journal of Global Optimization* 11 No. 4 (December 1997):341-359.

Subotic, Milos. "Artificial Bee Colony Algorithm for Constrained Optimization Problems Modified with Multiple Onlookers." *International Journal of Mathematical Models and Methods in Applied Sciences* Vol 6 No. 2 (2012): 314-322.

Tang, Jun and Xiaojuan Zhao. "A Hybrid Particle Swarm Optimization with Adaptive Local Search." *Journal of Networks* Vol. 5 No. 4 (April 2010):411-418.

Tang, K., X. Yao, P. N. Suganthan, C. MacNish, Y. P. Chen, C. M. Chen, and Z. Yang. *Benchmark Functions for the CEC'2008 Special Session and Competition on Large Scale Global Optimization*. Technical Report, Nature Inspired Computation and Applications Laboratory, USTC, China, 2007.

Tankasala, Ganga Reddy. "Artificial Bee Colony Optimisation for Economic Load Dispatch of a Modern Power System." *International Journal of Scientific and Engineering Research* Vol 3 No. 1 (January 2012):1-6

Teodorovic, Dusan. "Bee Colony Optimization." Chapter 3 in, *Innovations in Swarm Intelligence*. Chee Peng Lim, Lakhmi C. Jain, Satchidananda Dehuri Editors. Studies in Computational Intelligence Vol. 248. Springer-Verlag: Berlin, Germany. 2009. pages 39-60.

Tereshko, Valery and Andreas Loengarov. "Collective Decision-Making in Honey Bee Foraging Dynamics." *Journal of Computing and Information Systems* Vol. 9 No. 3 (2005):1-7.

Tuba, Milan, Nebojsa Bacanin and Nadezca Stanarevic. "Adjusted Artificial Bee Colony (ABC) Algorithm for Engineering Problems." *WSEAS Transactions on Computers* Vol 11 No. 4 (April 2012): 111-120.

Tukey, John W., *Exploratory Data Analysis*. New York: Addison-Wesley Publishing Company, 1977. 688 pages.

Uy, Nguyen Quang, Nguyen Xuan Hoai, RI McKay, and Pham Minh Tuan. "Initializing PSO with Randomised Low-Discrepancy Sequences: The Comparative Results." Pages 1985-1992 in, Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2007, 25-28 September 2007, Singapore. 2007.

Uyar, A. Sima and Belgin Turkay. "Evolutionary Algorithms for the Unit Commitment Problem." *Turkish Journal of Electrical Engineering and Computer Science* Vol.16 No. 3 (2008):239-255.

Valle, Yamille del, Ganesh Kumar Venayagamoorthy, Salman Mohagheghi, Jean-Carlos Hernandez and Ronald G. Harley. "Particle Swarm Optimization: Basic Concepts, Variants and Applications in Power Systems." *IEEE Transactions on Evolutionary Computation* Vol 12 No. 2 (April 2008):171-195.

Veselka, Thomas D., Leslie D. Poch, Clayton S. Palmer, Samuel Loftin and Brent Osiek *Financial Analysis of Experimental Releases Conducted at Glen Canyon Dam during Water Years 1997 through 2005* ANL/DIS-10-7 August 2010. [REVISED VERSION]

Veselka, Thomas D., Leslie D. Poch, Clayton S. Palmer, Samuel Loftin and Brent Osiek. *Ex Post Power Economic Analysis of Record of Decision Operational Restrictions at Glen Canyon Dam* ANL/DIS-10-6. Argonne National Laboratory. Argonne, Illinois. April 2010. 96 pages.

Veselka, Thomas D., O. Benjamin Schoepfle and Matthew Mahalik. *CRSP Basin-Wide Economic Methodology—Modeling the Aspinall Cascade*. Systems Science Group, Argonne National Laboratory. Argonne, Illinois: Argonne National Laboratory. July 2003.

Vicaria, Fernando. "Totally Random—Getting a Truly Random Number." *Delphi Informant* 9. No. 10 (October 2003):8-14.

Wahde, Mattias. *Biologically Inspired Optimization Methods-- An Introduction* Southampton, MA: WIT Press. 2008. 225 pages.

Weber, Ernst Juerg. "Optimal Control Theory for Undergraduates Using the Microsoft Excel Solver Tool." *Computers in Higher Education Economics Review (Cheer)* 19 No. 1 (2007):4-15.

Weise, Thomas. *Global Optimization—Theory and Practice* 2nd Edition. eBook. Version 2008-07-07. Available from: www.it-weise.de. 2008. 703 pages.

Wong, Tien-Tsin, Wai-Shing Luk and Pheng-Ann Heng. "Sampling with Hammersley and Halton Points." *Journal of Graphics Tools* 2 No. 2 (November 1997): 9-24.

Wolpert, David H. and William G. Macready. "No Free Lunch Theorems for Optimization" *IEEE Transactions on Evolutionary Computation* Vol. 1 No. 1 (April 1997):67-82.

Wood, Allen J. and Bruce F. Wollenberg. *Power Generation, Operation and Control.* 2nd Edition. New York, New York: John Wiley and Sons, 1996.

Wright, Alden H. "Genetic Algorithms for Real Parameter Optimization." In, *Foundations of Genetic Algorithms.* Gregory J.E. Rawlins (editor). San Mateo, CA: Morgan Kaufmann. 1991. pp. 205-218.

Wu, Bin and Cun Hua Quan. "Differential Bee Colony Algorithm for Global Numerical Optimization." *Journal of Computers* Vol. 6 No. 5 (May 2011): 841-848.

Yan, Yiming, Ye Zhang and Fegjiao Gao. "Dynamic Artificial Bee Colony Algorithm for Multi-Parameters Optimization of Support Vector Machine-Based Soft-Margin Classifier." *EURASIP Journal on Advances in Signal Processing* Vol 1 No. 146 (July 2012):1-13.

Yang, Xin-She. "Firefly Algorithm, Stochastic Test Functions and Design Optimization." *International Journal of Bio-Inspired Computation* Vol 2 No. 2 (March 2010):78-84.

Yang, Xin-She and Suash Deb. "Engineering Optimization by Cuckoo Search." *International Journal of Mathematical Modeling and Numerical Optimization.* Vol. 1 No.4 (April 2010):330-343.

Yang, Xin-She and Suash Deb. "Cuckoo Search via Levy Flights." in, *Proceedings of the World Congress on Nature and Biologically Inspired Computing* (NaBIC 2009, India). IEEE Publications, USA. pages 210-214.

Yang, Xin-She. "Engineering Optimizations via Nature-Inspired Virtual Bee Algorithms." in, *Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach.* Lecture Notes in Computer Science Vol. 3562. Springer: Berlin/Heidelberg. 2005. pages 317-323.

Zielinski, Karin and Rainer Laur. "Stopping Criteria for Differential Evolution in Constrained Single-Objective Optimizations." Chapter 5 in, *Advances in Differential Evolution.* Studies in Computational Intelligence, Volume 143. Edited by Uday K. Chakraborty. Springer-Verlang: Belin, Germany. 2008.

Zielinski, Karin, Petra Weitkemper, Rainer Laur and Karl-Dirk Kammeyer. "Examination of Stopping Criteria for Differential Evolution based on a Power Allocation Problem." In, volume 3, pages 149–156, Proceedings of the 10th International Conference on Optimization of Electrical and Electronic Equipment. Brasov, Romania, 18-19 May 2006.

Zielinski, Karin and Rainer Laur. "Stopping Criteria for Constrained Single-Objective Particle Swarm Algorithm." *Informatica* 31 No. 1 (March 2007):51-59.

Appendix 1. ABCO Algorithm

Introduction

Artificial bee colony optimization (ABCO) algorithms are among the most recently (circa 2001) described algorithms to appear in the literature. ABCO is an optimization approach based on the collective behaviour of a hive of honey bees, searching for food. ABCO utilizes employed bees and unemployed bees, composed of onlooker bees and scouts, to locate the best food sources in the search space, thereby identifying the optima of a function.

There are several algorithms related to honey bees and their behaviour and this can prove to be quite confusing. Literature examples include the bee algorithm (BA) proposed by Pham et al (2006), the virtual bee algorithm (VBA) used by Yang (2006), the bee colony optimization (BCO) algorithm of Lucic and Todorovic (2003), the honey bee mating optimization (HBMO) algorithm of Haddad, Ashfar and Marino (2006) and the artificial bee colony optimization (ABCO) algorithm. Given the rapid emergence of new bee-related algorithms, there can be no guarantee this list is exhaustive. Potentially, other bee related algorithms now exist that have not been identified by the author.

The ABCO algorithm was selected for use in this research. The purpose of this research project is relatively narrow—the application of promising algorithms to hydropower optimization problems, which are continuous nonlinear constrained optimization problems. Some of the bee-related algorithms are limited to discrete problem applications and others are either unsuitable for constrained continuous optimization problems, or have not been applied in this context. Based on the assessment by Mezura-Montes and Cetina Dominguez (2012), the artificial bee colony optimization (ABCO) algorithm is the most promising algorithm for this research application.

Karaboga et al (2012) attribute the first description of the ABCO algorithm to a seminal paper by Karaboga (2005). Karaboga (2005), in turn, graciously credits a paper by Tereshko and Loengarov (2005) for inspiration. Since the mid 2000's, ABCO has been applied in a number of disciplines and optimization contexts. Excellent surveys of these applications can be found in Teodorovic (2009), Kaur and Goyal (2011), Teodorovic, Davidovic and Selmic (2011) and Karaboga et al (2012).

Several published ABCO applications are of significance to this research effort. Sharma, Pant and Singh (2012) demonstrate the use of the ABCO algorithm for solving selected engineering design problems and introduce some enhancements

aimed at improving the convergence speed and exploitation capability of the algorithm. Research reported by Stanarevic, Tuba and Bacanin (2011), Karaboga and Akay (2011) and Brajevic and Tuba (2012) advance the application of the basic ABCO algorithm and successfully apply it to constrained optimization problems. Mezura-Montes and Cetina-Dominguez (2012) describe a systematic empirical analysis of a modified ABCO algorithm for constrained optimization. Of particular saliency to this effort are the Hemarmalini and Simon (2011), Sharma, Jain and Pandit (2011) and the Tankasala (2012) applications of ABCO to the economic dispatch problem, the subject of the Phase 1 research effort (Harpman 2012).

ABCO Terms

There are several ABCO specific terms commonly used in the literature. Their usage varies somewhat from the usage in other evolutionary algorithms reviewed previously. It is likely that these terms are used differentially in various ABCO code implications. The following terms are defined consistently with the ABCO Delphi code developed by Karaboga et al (2012) and labeled as version 5/17/2011 (http://mf.erciyes.edu.tr/abc/). It is useful to note this code was explicitly developed for function minimization.

- Fitness function- transformed[4] objective function value.
- Fitness- value of the fitness function
- Penalty function—a function which assigns a value to each constraint violations.
- Penalty—the aggregate penalty value of constraint violations.
- Population best – best fitness achieved by any individual in the population

Individual Components

In the context of the ABCO algorithm, each individual is called a bee (of course!). There are employed bees and unemployed bees. The unemployed bees are of two types; onlooker bees and scout bees.

Each of the np individual bees in the virtual population consists of the following components, where d is the number of dimensions in the problem:

- Coordinates of its current position: $x=(x_1 \ldots x_d)$
- Fitness
- Penalty

[4] See equation (11) for the details of this transformation.

In the ABCO Delphi code developed by Karaboga (http://mf.erciyes.edu.tr/abc/), scalars, vectors and matrices, but not data structures or objects, are employed. For consistency and to allow for selective code reuse, the same approach is used in this research project. Conversely, in the ABCO C# code described in Bacanin, Tuba and Brajevic (2011) it appears that each individual bee is coded as an object. This programming approach facilitates multi-threaded and parallel computing options. The object oriented approach would appear to be programmatically more efficient, but certainly requires a higher degree of programming skill and effort to implement.

Basic ABCO Algorithm

The basic ABCO algorithm is relatively straightforward as illustrated in Figure 20. First, each of the np members of the bee population is created, their positions initialized in the search space and their fitness evaluated and stored. The ABCO iterative process, which in this case is termed "cycling", then begins. During each iteration or cycle, there are two population subsets, employed bees and unemployed bees. Employed bees search for new food sources within the fitness landscape. The employed bees then return to the hive where they share the information they have gathered about food sources with the unemployed bees.

Unemployed bees consist of two types; onlooker bees and scout bees. The employed bees share the fitness and location information they have gathered with the onlooker bees back at the hive. The onlooker bees probabilistically choose this information. The onlooker bees fly out from the hive to the chosen location and search in the neighborhood for an improved solution. If the fitness of a particular employed bee does not improve in a preset number of cycles, it becomes a scout bee which is reinitialized randomly in the search domain.

At the end of each cycle, a test is applied to determine if the bee population has converged. If the population has converged, the iterative process is terminated and the results are reported. If the population has not converged, a new iteration is undertaken. This process continues until the either the population has converged or the maximum number of iterations has been completed.

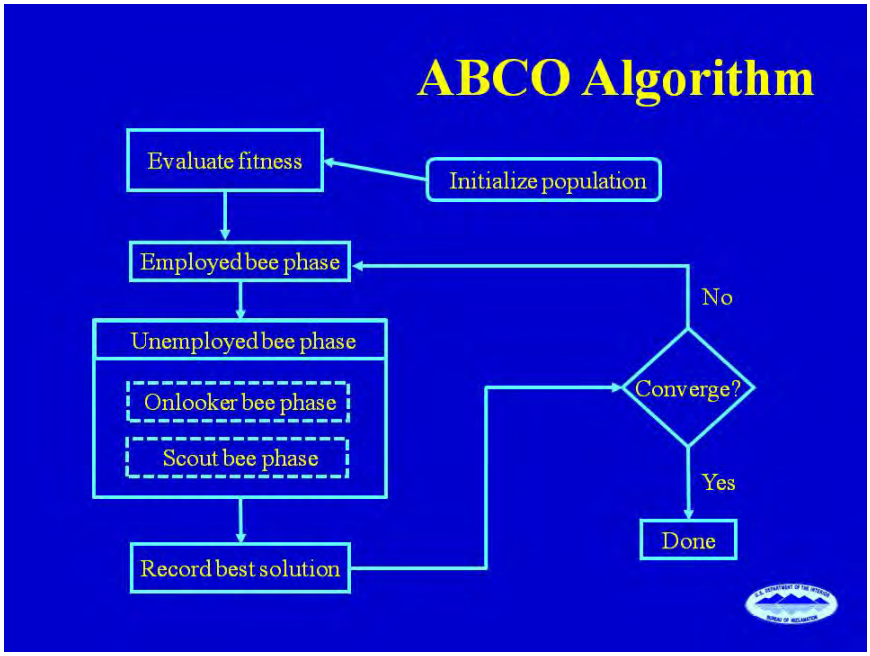

Figure 20.—Illustration of the Basic ABCO algorithm.

Initialization

The ABCO algorithm developed by Karaboga et al (2005), like the vast majority of heuristic algorithms, utilizes a randomized approach for determining the starting positions of all np-bees within the d-dimensional search space. This process is termed, "initialization."

As this concept is carried out, for each bee (i) in the populations, an initial position in the search space is assigned by using expression (10)

(10) $$x_{i,j} = l_j + rand_j * (u_j - l_j)$$

Where: $x_{i,j}$ = value of decision variable (j) for bee (i)
l_j = lower bound value for variable (j)
u_j = upper bound value for variable (j)
$rand_j$ = uniform random variable (0,1).

The vast majority of applied work uses the uniform random distribution to initially locate points in the search space. Although easily implemented, the weakness of this approach is that a non-systematic location of the initialized points can result in the space. As discussed in the Phase I Report (Harpman 2012, p. 25-27), a frequent contributor to this poor performance is the quality of the random number generator (RNG). Appendix 12 in Harpman (2012) contains a rather exhaustive treatment of this topic and reports on the RNG selected for use

in this research effort. Low discrepancy sequences and other approaches are sometimes employed to overcome this behavior. A useful treatment of these topics, and an explanation of their application in this research, may also be found in the Phase I Report (Harpman 2013, Appendix 13).

Fitness Evaluation

Fitness evaluation in the ABCO algorithm differs somewhat from the approach used in the PSO, DE and RCG algorithms. In the latter cases, the fitness value is composed of the objective function value plus the value of the penalty, if any. For the ABCO algorithm however, the fitness is composed solely of the objective function value, mathematically transformed as shown by equation (11).

(11), $$fit(x) = \begin{cases} \dfrac{1}{1+f(x)} & if \;\; f(x) \geq 0 \\ 1+abs(f(x)) & if \;\; f(x) < 0 \end{cases}$$

The ABCO algorithm of Karaboga et al (2005) is designed for function minimization and it tracks the value of the penalty separately from the fitness value.

Employed Bees Phase

Employed bees search the fitness landscape within the neighborhood of their position for new locations which have more nectar (are more fit). Each employed bee finds a neighboring food source and then evaluates its fitness. Although there are variations on this approach, the most common approach is to generate a neighboring position using the formula shown in equation (12).

(12) $$y_{i,j} = x_{i,j} + Urand_j * (x_{i,j} - x_{k,j})$$

Where: $y_{i,j}$ = value of neighborhood food source variable (j) for bee (i)
$x_{i,j}$ = value of decision variable (j) for bee (i)
$Urand_j$ = uniform random variable (-1,1).
$x_{k,j}$ = value of decision variable (j) for randomly chosen bee ($k \neq i$)

Using expression (12), a neighborhood location is chosen by first randomly selecting another employed bee. Then, for each dimension, subtracting its location from the current location of the employed bee, multiplying the resulting difference by a random value drawn from the range [-1,+1] and adding the result to the original value. This process yields a new location, specified in d-dimensions, in the neighborhood of the employed bee.

One aspect of relationship (12) could be overlooked but is useful to understand. As the difference between the x-values of the employed bee and a randomly

chosen bee, decrease, the perturbation also decreases. Thus, as the search approaches the optimum solution in the search domain, the step length also decreases.

The objective function value of this neighborhood location and its fitness value are then calculated as described by equation (11).

The fitness of the neighborhood location and the original location of the employed bee are compared using a so-called "greedy" selection mechanism. The employed bee either moves to the neighboring location, if this move is fitness improving, or it maintains its current location and fitness. This decision is based on the relationship shown in (13).

(13) $$P_{i,t+1} = \begin{cases} p_i & if \;\; p_i \geq d_i \\ d_i & if \;\; d_i > p_i \end{cases}$$

Where: $P_{i,t+1}$ = employed bee's next position.
p_i = employed bee's initial position and fitness
d_i = neighborhood position and fitness.

Unemployed Bees Phase

After searching their neighborhoods for more promising sources of nectar, the employed bees return to the hive and share their knowledge of the fitness landscape with the unemployed bees. There are two types of unemployed bees; onlooker bees and scouts.

Onlooker Bees

The employed bees communicate with the onlooker bees and convey information about the locations they have visited and the nectar content (or fitness) at those coordinates in the d-dimensional search space. The waiting onlooker bees probabilistically choose the food sources they will visit. In the ABCO algorithm they choose a food source depending on the rationalized probability values associated with these sources. In the Karaboga et al (2005) algorithm, roulette wheel selection is employed for this purpose.

Using the roulette wheel selection approach, the probability that an onlooker bee will choose the location of any particular employed bee is given by *ProbC*, which can be calculated using the expression given by (14).

(14) $$ProbC = \frac{fit_i}{\sum_{i=1}^{np} fit_i}$$

The roulette wheel selection process is frequently employed in genetic algorithms. An excellent description of the roulette wheel selection process and an example of its application is provided by Haupt and Haupt (2004).

After probabilistically selecting a promising food source to visit, the onlooker bees fly out into the fitness landscape to this location. The bees then began searching in the neighborhood of this location for new, better sources of nectar. Each bee generates a neighboring position and evaluates the fitness at this neighboring location using the relationships previously shown in equations (11) and (12). The fitness of the neighborhood location and the original location of the onlooker bee are compared using the selection mechanism described by equation (13). The onlooker bee either moves to the neighboring location, if this move is fitness improving, or it maintains its current location and fitness. At the conclusion of the unemployed-onlooker bee phase, the global best position and fitness are recorded, prior to starting a new cycle or iteration of the algorithm.

Scout Bees

Employed bees whose fitness cannot be improved within some preset number of cycles (iterations) specified by the analyst become scout bees. Scout bees abandon their food sources (locations) and are randomly reinitialized in the fitness landscape. These scout bees begin searching for new, superior nectar sources at new locations. Through this interesting mechanism, initially discovered inferior fitness locations and high value but stagnant locations are abandoned. This allows new, potentially superior positions to be explored. The conversion of some of the employed bees to scouts helps preserve fitness exploiting behavior and reduce the likelihood of premature convergence on a local optimal point. At the conclusion of the unemployed-scout bee phase, the global best position and fitness are recorded, prior to starting a new cycle or iteration of the algorithm.

Convergence

The preponderance of numerical optimization algorithms are based on some sort of iterative or repetitive procedure. An important aspect of these algorithms is the design of intelligent convergence or stopping rules. These rules detect when the routines have converged on a solution, and then halt the iterative process.

The ABCO algorithm developed by Karaboga et al (2012) and typified by the Delphi code example, dated 5/17/2011 (http://mf.erciyes.edu.tr/abc/), uses a preset maximum number of cycles (iterations) as a stopping rule. In this respect, this example of the ABCO algorithm is identical to the vast majority of heuristic algorithms. Using this approach, the algorithm proceeds until a pre-set number of iterations have been completed-- then it halts. The "best" solution from the population of solutions is identified and then reported. The primary advantage of this approach is it is simple to implement. This stopping rule is frequently used to compare the behavior of alternative parameter settings and algorithm variants. The disadvantage is profound—the preset maximum number of iterations may or

may not correspond to the number of iterations required for algorithm convergence.

The Phase I Research Report contains a rather exhaustive treatment of evolutionary algorithm stopping rules. It introduces several improved convergence approaches and describes their application to DE, PSO and RCGA, in replicated experiments (Harpman 2012, pages 35-39 and pages 54-58). Conceptually, similar performance improvements could be achieved by integrating some of these convergence approaches with the ABCO algorithm.

Appendix 2. ABCO Parent Code Base

The artificial bee colony (ABCO) optimization algorithm is a recent innovation by Karaboga (2005). There are a number of descriptions of this algorithm and these vary considerably in terms of scope and detail. Some of these descriptions include author specific embellishments and in some offerings, descriptions of the basic approach appear to vary significantly from the original source (Karaboga 2005). For efficiency reasons, the parent Delphi code base for the ABC optimization algorithm used in this effort was downloaded directly from the Artificial Bee Colony Algorithm website homepage (http://mf.erciyes.edu.tr/abc/). Posted as version 5/17/2011, this Delphi code was designed for constrained minimization and includes several example engineering problems which are described in Akay and Karaboga (forthcoming), Akay and Karaboga (2011) and Karaboga and Basturk (2007).

In contrast to some previous adventures, the Delphi code obtained from the ABC algorithm page compiled flawlessly in Borland Studio Developer 2006 and appeared to function correctly. Using this parent code as a starting point, some new modular and more portable Delphi code was developed. The improved ABC code was further revised, rewritten and applied to the suite of three unconstrained maximization test problems examined in Phase I of this research project. The resulting code was tested and validated for subsequent use in this project.

Appendix 3. Optimization, Minimization and Maximization.

Many optimization algorithms are designed to identify the minima of a function. Similarly, the vast majority of explanations and documentation for optimization routines are couched in terms of function minimization (for example, see Press, et al 1989). In the case examined in this document however, the focus is on maximization. Luckily, there is a straightforward relationship between identifying the minima of a function and finding the maxima of a function. This relationship can and is exploited to use algorithms designed for minimization on maximization problems.

As described in Press et al (1989), the tasks of minimization and maximization are "trivially related to each other." Specifically, the minima of an arbitrary function, f(x) is the maxima of the negative of the function, –f(x). As with many purportedly trivial mathematical concepts, understanding is greatly facilitated by a *simple* 2-dimensional example.

Consider the sphere function in 2-dimensions[5]. The sphere function is written as shown in equation (5).

(15) $$Z = -(x^2).$$

Graphing the function described by equation (15) in 2-dimensions (x, y space) for x over values ranging from –6.0 to +6.0 yields the plot shown on the bottom half of Figure 21 as an inverted bowl shape in blue.

The negative of the function specified by equation (15) is written as shown in in equation (16).

(16) $$-Z = (x^2)$$

Graphing equation (16), which again is the negative of equation (15), in 2-dimensions for x ranging from –6.0 to +6.0 yields the plot shown in upper half of Figure 21, as a red, bowl shaped surface.

[5] Note the 3-dimensional sphere function was used as an algorithm testing and development function in Phase 1 of this research project. By extension, this makes it an obvious choice for this illustrative example.

Note that in Figure 21 the plot of Z is the mirror image of the plot of –Z, and vice versa. The optimal point (Z*), the maxima in the case of equation (16) and the minima in the case of equation (15) is identical (Z*=0.0) and both occur when the value of x=0.0.

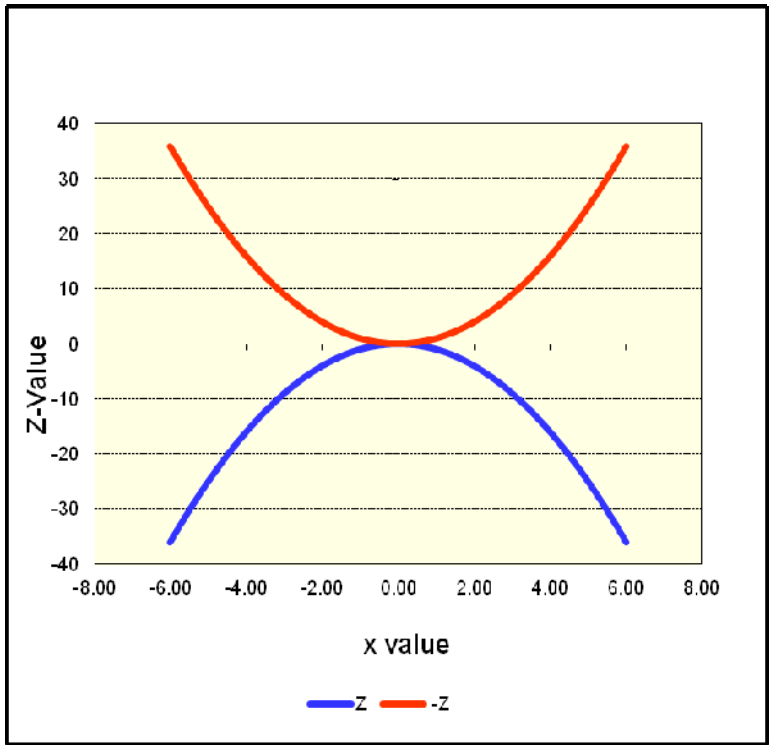

Figure 21.—Minimizing –f(x) is Identical to Maximizing f(x).

Appendix 4. Test Functions for Algorithm Development

Introduction

This Appendix describes three optimization test functions which were employed in the early phases of algorithm development, including coding, testing, performance visualization and validation. Each of these functions is a previously studied 3-dimensional (3-D) continuous, unconstrained, optimization problem. The test functions selected were restricted to three dimensions to facilitate implementation and to allow for real-time visualization of the algorithm's behavior in the search space. These test functions facilitated development of the evolutionary algorithms, prior to their application to the more difficult and higher dimension electric power-related problems, which were the focus of this research.

By design, this research effort utilizes only a small and rather rudimentary subset of the universe of test functions investigated by other authors. As popularized by De Jong (1975), many existing studies examine the performance of evolutionary algorithms on a suite of optimization test functions (for example, see Mishra 2007 or Mezura-Montes and Flores-Mendoza 2009). There are numerous optimization test functions available for this purpose, some of which are exceedingly complex. A sample of the test functions encountered in this literature is described in De Jong (1975), Haupt and Haupt (2004), Engelbrecht (2005), Price, Storn and Lampinen (2005), Feoktistov (2006) and other sources.

Test Function 1—Sphere

The sphere function is one of the most rudimentary 3-D optimization problems. It is a symmetric, continuous real-valued function possessing a single (global) optimal point. This function is defined over the set of all real numbers, however a bounded search range is used in this application.

In 3-D, the equation describing the sphere function is (17).

$$Z = -x^2 - y^2 \tag{17}$$

The gradient of this test function is especially useful as a device for ascertaining the quality of a solution on convergence. For this test function, the expression for the gradient, or vector of first partial derivatives, is shown in equation (18).

$$\nabla Z_x = \frac{\partial Z}{\partial x_i} = -2x_i \tag{18}$$

The domain for the sphere function is the real number line ($-\infty \leq x \leq +\infty$). For test purposes however, the search domain for the independent variables (x, y) was restricted to the bounded interval ($-2 \leq x \leq +2$).

The maximum value of Z for this test optimization function is Z=0.0. This maximum Z value is obtained when x=0.0 and y=0.0.

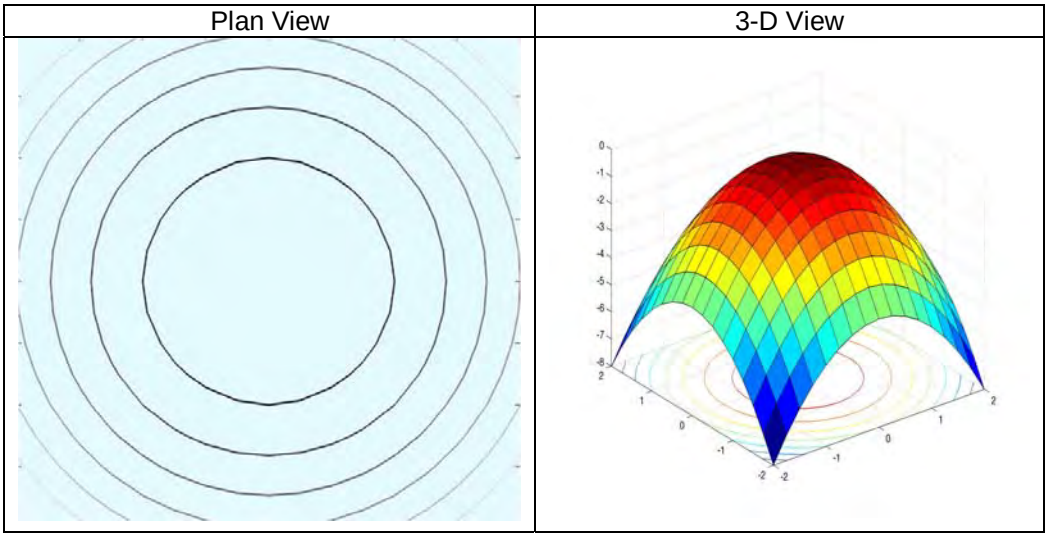

Figure 22.—Plan and 3-D Views of the Sphere Function.

Figure 22 illustrates the plan (top) view and the 3-D view of test function 1, the sphere function. As shown in this figure, the contours are symmetric about the optimal point. The global maximum is the sole optima in the bounded search space.

The sphere function is perhaps the most rudimentary of all optimization test problems. It is symmetric about the origin, easily implemented in code and readily solved. This function was used primarily for early-stage development of the evolutionary algorithms employed in this research. This test function allowed for visual verification of algorithm functioning and effectiveness during the coding process.

Test Function 2—Ridge

The ridge function is a somewhat more complex 3-D optimization problem. It is a continuous but not a symmetric function. It has a single (global) optimal point located at the top of a ridge, bounded on either side by steep canyons. This function is defined over the set of all real numbers, however a bounded search range is used in this application.

In 3-D, the equation describing the ridge function is (19).

(19) $$Z = x + 2ey - e^{x} - e^{2y}$$

The gradient of this test function is especially useful as a device for ascertaining the quality of a solution on convergence. For this test function, the expressions for the gradient, or vector of first partial derivatives, is shown in equations (20) and (21)

(20) $$\nabla Z_x = \frac{\partial Z}{\partial x} = 1.00 - e^{x}$$

(21) $$\nabla Z_y = \frac{\partial Z}{\partial y} = 2e - 2e^{2y}$$

The domain for the ridge function is the real number line ($-\infty \leq x \leq +\infty$). For test purposes however, the search domain for the independent variables (x, y) was restricted to the bounded interval ($-2 \leq x \leq +2$).

The maximum value of Z for this test optimization function is Z=-1.00. This maximum Z value is obtained when x=0.0 and y=0.50.

Figure 23 illustrates the plan (top) view and the 3-D view of test function 2, the so called ridge function. As shown in this figure, the contours are asymmetric about the optimal point. The global maximum is the sole optima in the bounded search space. It lies at the top of a long gently sloping ridge with canyons on either side.

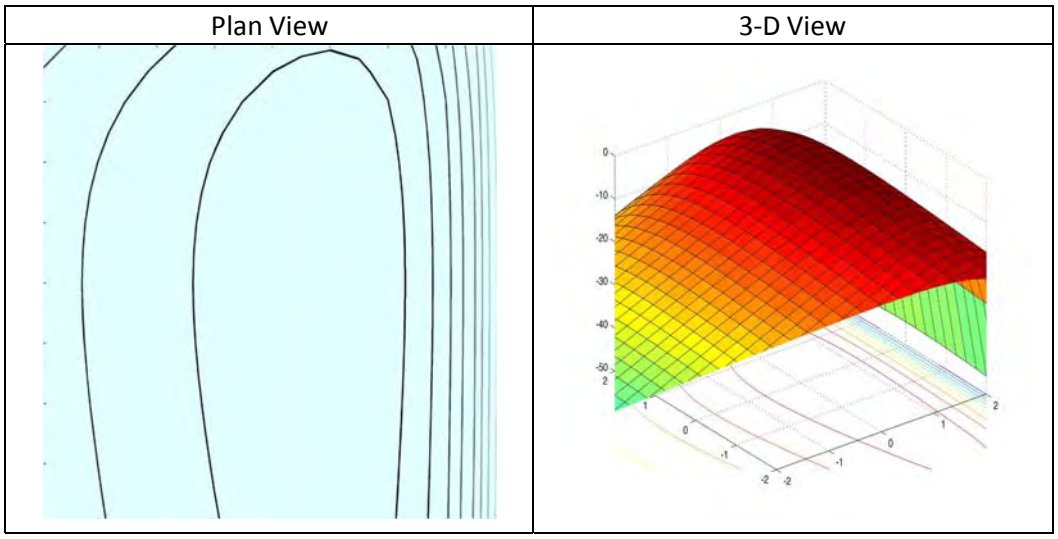

Figure 23.—Plan and 3-D Views of the Ridge Function.

The slope of the ridge changes very gradually, often causing premature convergence for certain types of gradient based algorithms and poorly parameterized evolutionary algorithms. A single badly calculated step can send

the solution down one of the precipitous canyons on either side of the ridge, causing the algorithm to fail.

In the general scheme of things, the ridge function is a relatively straightforward optimization test problem. It is easily implemented in code, although not so readily solved. This function was used as a test bed during the development of the evolutionary algorithms described in this research. This test function allowed for visual verification of algorithm functioning and effectiveness during the coding process.

Test Function 3—Alpine

The Alpine test function, as described by Clerc (2006) and Haupt and Haupt (2004), is a complex 3-D optimization problem. It is a continuous, but not symmetric function. It has a multiple local optima and a single (global) optimal point in the search space. This function is defined over the set of all real numbers and has multiple local optima in that range (as might be expected). A finite bounded search range is used in this application.

In 3-D, the equation describing the Alpine function is (22).

$$Z = \sin(x) \times \sin(y) * \sqrt{xy} \tag{22}$$

The gradient of this test function is especially useful as a device for ascertaining the quality of a solution at convergence. For this test function, the expressions for the gradient, or vector of first partial derivatives, is shown in equation (23).

$$\nabla Z_x = \frac{\partial Z}{\partial x} = \tanh(x_i) + 2x_i \tag{23}$$

The domain for the ridge function is the real number line ($-\infty \leq x \leq +\infty$). For test purposes however, the search domain for the independent variables (x, y) was restricted to the bounded interval ($-10 \leq x \leq +10$).

The maximum value of Z for this test optimization function is Z=7.885600724. The maximum Z value is obtained when x=7.917052686 and y=7.917052686. This point is located in the upper right-hand quadrant of the plot.

Figure 24 illustrates the plan (top) view and the 3-D view of test function 3, the Alpine function. As shown in this figure, the contours are quite complex. There are multiple local optima in the search space. The global maximum (in this bounded search space) is located at the top of Mount Blanc (as termed by Clerc 2006) or Longs Peak (as termed by Haupt and Haupt 2004), in the upper right-hand quadrant of the plot. Mount Blanc is surrounded by lesser peaks. Encountering any of these lesser peaks will cause a gradient based algorithm to converge and announce it has located a solution. Identification of the global optima in this search space is extremely difficult.

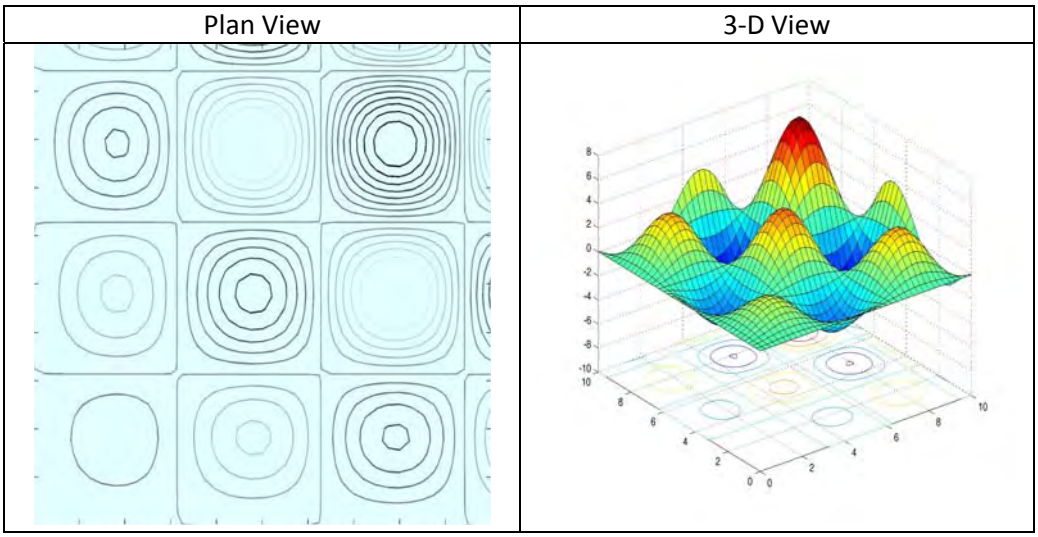

Figure 24.—Plan and 3-D Views of the Alpine Function.

In the general scheme of things, the Alpine function is a relatively complicated optimization test problem. It is quite easily implemented in code, although not so readily solved, particularly by gradient based methods. This function was used extensively during the development of the evolutionary algorithms described in this research. This test function allowed for visual verification of algorithm functioning and effectiveness in a complicated solution space.

Appendix 5. Reservoir Characteristics

Operation of the hydropower plant is made possible by and affects the state of the storage reservoir. Conversely, the state of the storage reservoir, in particular its elevation and the existence of physical and engineering constraints directly effects the manner in which the hydropower plant can be operated.

The reservoir characterized in the ESIM03 model has a maximum storage capability of 640,000 af when it is full. A reservoir full condition occurs when the water surface elevation reaches 2008.1856 ft above mean sea level. The reservoir can be drafted by 200 ft for power generation purposes. The minimum power pool occurs at an elevation of 1808.1856 ft or a volume of 240,795.83 af. There is a difference between the elevation at the top of the penstock and the minimum power pool elevation. This difference reflects the minimum submergence depth[6] necessary for power production. When there is no release from the dam, the elevation of the tailwater is 1708.1856 ft. An illustration of these critical elevations and depths is provided in Figure 25.

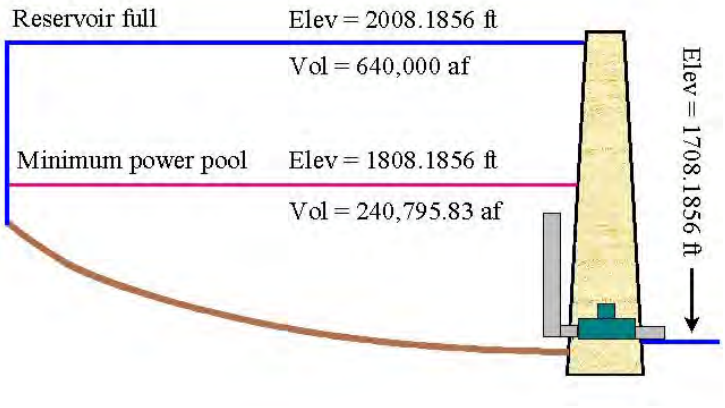

Figure 25.—Critical Reservoir and Powerplant Elevations.

The relationship between the volume content of the reservoir and the elevation of the reservoir is represented by a cubic polynomial of the form shown in equation (24).

(24) $$elev = a \times vol^3 + b \times vol^2 + c \times vol + d$$

Where: elev=water surface elevation of reservoir (ft above mean sea level)
vol=volume of the reservoir (acre-feet)

[6] A minimum submergence depth is necessary to prevent the creation of vortices and the entrainment of air in the turbines. If this were to occur, it would cause cavitation and damage the turbines.

a, b,c,d are coefficients.

The coefficients for this relationship are shown in Table 11.

Table 11.—Coefficients for Volume and Elevation Equation

Coefficient	Value
a	2.400e-15
b	-3.85e-09
c	0.0024
d	1420.00

Figure 26 shows a plot of this relationship (equation 24) over a range of applicable storage volumes.

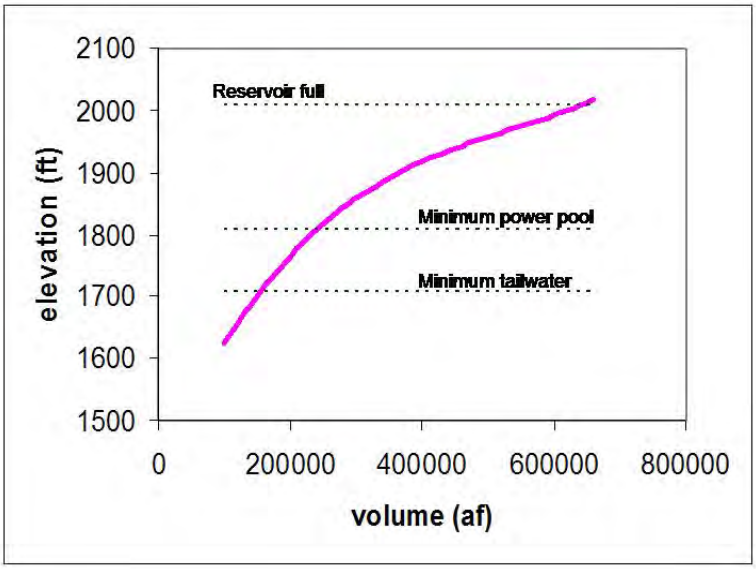

Figure 26.—Relationship Between Reservoir Volume (af) and Elevation (ft).

The relationship shown characterizes a topography which is fairly typical for storage reservoirs. Initially, the elevation increases rapidly as the volume of the reservoir increases. As the reservoir approaches its full condition, the elevation increases much less rapidly.

There is a one to one relationship between reservoir volume and reservoir surface elevation. Specifying the reservoir volume is equivalent to specifying the elevation and *vice versa.* Since some calculations require the elevation and some calculations require the reservoir volume, the ability to convert rapidly back and forth between the two measurements is essential. Given a specified reservoir volume, the reservoir elevation is easily computed by using equation (24). However, computing the reservoir volume from a known reservoir elevation requires the solution of a cubic equation which is somewhat more difficult. The (cubic) relationship for volume as a function of elevation can be solved using the methodology described in Press et al (1989 page 163).

Appendix 6. Dispatch Test Problems

Narrative Description

Each of the dispatch test problems represents a hypothetical hydropower plant with either two generators or four generators, and an outlet works. The generator units have different engineering and operational characteristics as described subsequently. The elevation of the storage reservoir (forebay) and the amount of water which must be released from the facility are assumed to be known (exogenously determined). The head is calculated as the difference between the reservoir elevation and the tailwater elevation. The elevation of the tailwater (also known as the afterbay or tailrace) is an increasing function of the total amount of water released from the plant. Each generator has a minimum and maximum release constraint. For any given reservoir elevation and release level, the objective of the operator is to release water from the operating generation units and the outlet works to maximize the total generation.

Mathematical Specification

The Test Dispatch Problems can be specified relatively efficiently in mathematical notation. This problem is a nonlinear mixed integer optimization problem with constraints. It can be written as shown in equations (25) though (30).

(25) $$Max \sum_{i=1}^{i=N} [p_i(q_i, Q)]$$

Subject to:

(26) $$\sum_{n=1}^{n=N+1} q_i = Q$$

(27) $$q_i \geq 0$$

(28) $$if\ q_i \neq 0,\ q_i \geq k_i$$

(29) $$q_i \leq w_i$$

(30) $1808.19 \leq elev \leq 2008.19$

where: i = source indicator
N = number of release sources (generators + outlet works)
p_i = real power generated (MW) at generator (i)
q_i = release of water (cfs) from source (i)
Q = total release from all sources (cfs)
elev = forebay reservoir elevation (ft above mean sea level)
k_i = minimum release constraint (cfs)
w_i = maximum release constraint (cfs)

Equation (25) is the objective function for this constrained maximization problem. The operator's objective is to maximize generation by optimally choosing the amount of water to release from the available N-generators, and the outlet works. Water released from the outlet works does not generate electricity but does affect the tailwater elevation and consequently the amount of electricity generated at the operating generation units.

Equation (26) is the applicable water balance constraint. This constraint ensures the amount of water released from all sources (generators 1 through N, and the outlet works) exactly equals the intended release, Q.

Equation (27) is a nonnegativity constraint. This logical constraint is needed to ensure that all of the releases, from all sources, are nonnegative.

Equation (28) is a deceptively complex logical and minimum release constraint. In narrative form, this equation requires that the release from a particular source, such as turbine 2, must be either 0.0 (off), or, if it is positive (on), it must be greater than some given minimum release constraint (k). This type of constraint is often coded as an binary (1/0) constraint and it vastly increases the complexity of this problem.

Equation (29) is the maximum release constraint. Releases from any source (i) must be less than or equal to the maximum physical release capabilities of that source (w).

Constraint equation (30) delineates the limits of the active pool in the hypothetical storage reservoir. Allowing for a minimum submergence depth, the lower limit for generation occurs at a reservoir elevation of 1808.19 ft. The normal full pool for this hypothetical storage reservoir occurs at a reservoir elevation of 2008.19 ft. Appendix 5 illustrates the characteristics of this hypothetical forebay storage reservoir.

Common Mathematical Structure

In this analysis, the real power generation for any particular generator (i) is characterized by the expression shown in (31).

$$p_i = \left(\frac{q_i \times \gamma \times eff_i \times head(Q, elev)}{fptohp} \right) \left(\frac{hptokw}{1000} \right) \tag{31}$$

Where:

p_i = (real) electric power generated (mw) by generator (i)
γ = 62.40, specific weight of water at 50 degrees Fahrenheit (lbs/ft^3).
eff_i= efficiency factor[18] (dimensionless) for generator (i)
q_i = release (cfs) from source (i)
Q = total release from all sources.
elev = reservoir elevation (ft above mean sea level).
head = (gross) head (ft)
fptohp = 550, foot-pounds/sec to horsepower conversion factor.
hptokw = 0.746, horsepower to kilowatts conversion factor.

For purpose of this problem, (gross) head is defined as the difference between the reservoir elevation and the tailwater elevation. While it is assumed the forebay reservoir elevation is known, the elevation of the afterbay or tailwater varies with the total release from all sources, Q. More explicitly, the head depends not only on the releases made from, for example turbine 1, but also on releases being made from the remaining turbines and the outlet works, if any. For our purposes, we characterize net head as shown in (32).

$$head = [elev - (w0 + w1 \times Q)] \tag{32}$$

Where:

head = generation head (ft)
elev = reservoir elevation (ft above mean sea level).
w0 = 1708.186, tailwater elevation (ft above mean sea level) when Q=0.0
w1 = 0.0070, change in tailwater elevation as release changes (ft/cfs)
Q = total release from all sources

[18] In this application the efficiency (eff) is represented as a constant. More generally, efficiency may vary as a function of release and head.

Prohibited Operating Zones

Prohibited operating zones are operational or production ranges which should be avoided for engineering and functional reasons. Operation of a unit within these prohibited zones may cause the machinery to vibrate excessively (rough zones), may cause cavitation, or may result in other undesirable system states.

In general prohibited operating zones are delimited with a lower and operating zone limit. For example, a prohibited operating zone may be described as the zone between a release of 1000 cfs and 3400 cfs, or, alternatively, the generation levels between 100 MW and 200 MW.

The locations of prohibited operating zones are dynamic. Stated differently, the lower and upper limits of prohibited operating zones, and the range between them, is a function of the head, as well as the engineering characteristics of the installed equipment.

Dispatch Test Problem 1

Dispatch Test Problem 1 has one small generation unit (Unit 1), with a nominal capacity of 64 MW, and one large generation unit (Unit 2), with a nominal capacity of 153 MW. The relationship between release and generation for both of these units is shown in Figure 27.

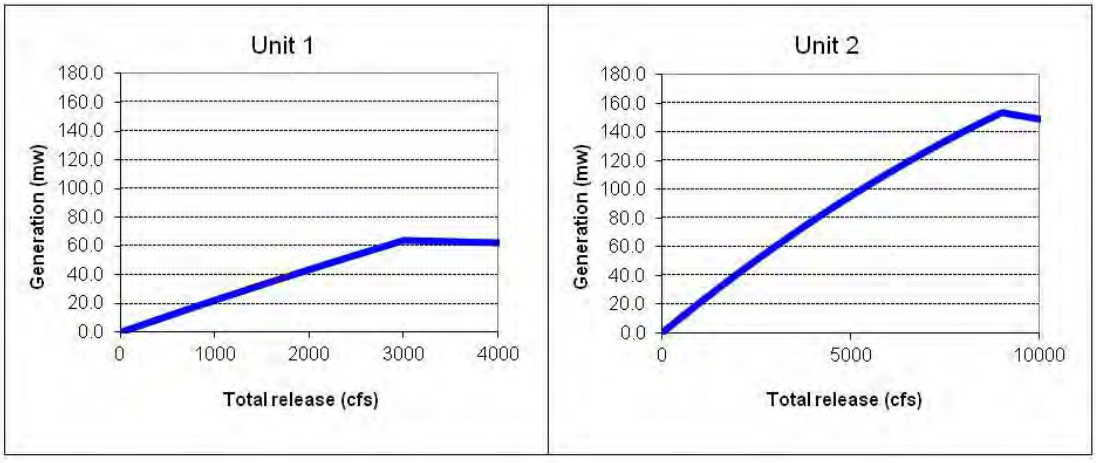

Figure 27.—Dispatch Problem 1 Release and Generation Characteristics.

Each of these generation units has minimum and maximum operational release capacities which are described in Table 12. For Unit 1, the minimum release capacity is 20 cfs and the maximum release capacity is 3000 cfs. In Test Problem 1, there are no prohibited operating zones for Unit 1. For Unit 2, the minimum

release capacity is 200 cfs and the maximum release capacity is 9000 cfs. In Test Problem 1, there are no prohibited operating zones for Unit 2. The release constraints for all of the units in Test Problem 1 are summarized in Table 12.

Table 12.—Dispatch Test Problem 1 Characteristics

	Release Constraints		Prohibited Zone 1		Prohibited Zone 2	
Unit	MinQ (cfs)	MaxQ (cfs)	LowerQ (cfs)	UpperQ (cfs)	LowerQ (cfs)	UpperQ (cfs)
1	20	3000				
2	200	9000				

Dispatch Test Problem 2

Dispatch Test Problem 2 has one small generation unit (Unit 1), with a nominal capacity of 64 MW, and one large generation unit (Unit 2), with a nominal capacity of 153 MW. The relationship between release and generation for both of these units is shown in Figure 28.

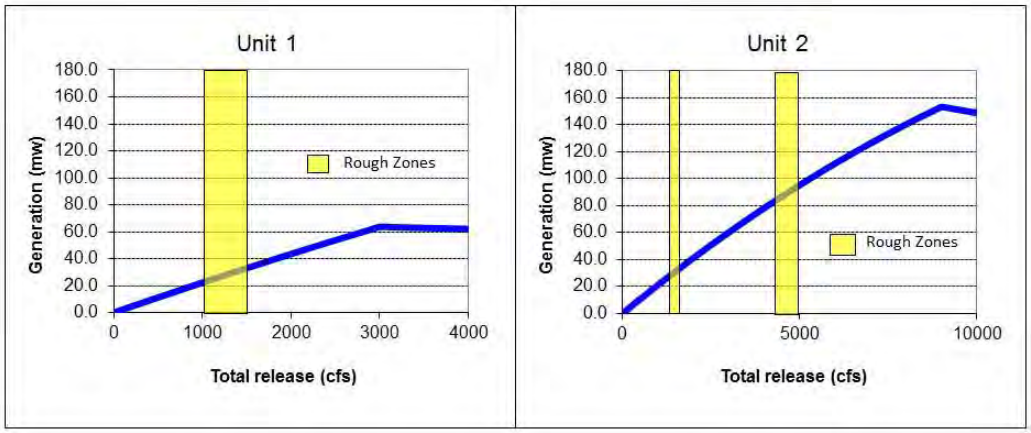

Figure 28.—Dispatch Problem 2 Release and Generation Characteristics.

Each of these generation units has minimum and maximum operational release capacities. For Unit 1, the minimum release capacity is 20 cfs and the maximum release capacity is 3000 cfs. In Test Problem 2, Unit 1 has a single prohibited operating zone. The lower boundary for prohibited operating zone 1 in Unit 1 is 1000 cfs. The upper boundary for prohibited operation zone 1 for this unit is 1500 cfs. For Unit 2, the minimum release capacity is 200 cfs and the maximum release capacity is 9000 cfs. In Test Problem 2, Unit 2 has two prohibited operating zones. The lower boundary for prohibited operating zone 1 is 1300 cfs.

The upper boundary for prohibited operation zone is 1700 cfs. For prohibited operating zone 2, the lower boundary is 4000 cfs. The upper boundary for prohibited operation zone 2 is 5000 cfs. The release constraints and prohibited operating zones for all of the units in Test Problem 2 are summarized in Table 13.

Table 13.—Dispatch Test Problem 2 Characteristics

	Release Constraints		Prohibited Zone 1		Prohibited Zone 2	
Unit	MinQ (cfs)	MaxQ (cfs)	LowerQ (cfs)	UpperQ (cfs)	LowerQ (cfs)	UpperQ (cfs)
1	20	3000	1000	1500		
2	200	9000	1300	1700	4000	5000

Dispatch Test Problem 3

Dispatch Test Problem 3 has four total generation units. These consist of two small generation units (Unit 1 and Unit 3), with a nominal capacity of 64 MW each, and two large generation units (Unit 2 and Unit 4), each of which has a nominal capacity of 153 MW. The relationship between release and generation for all of these units is shown in Figure 29.

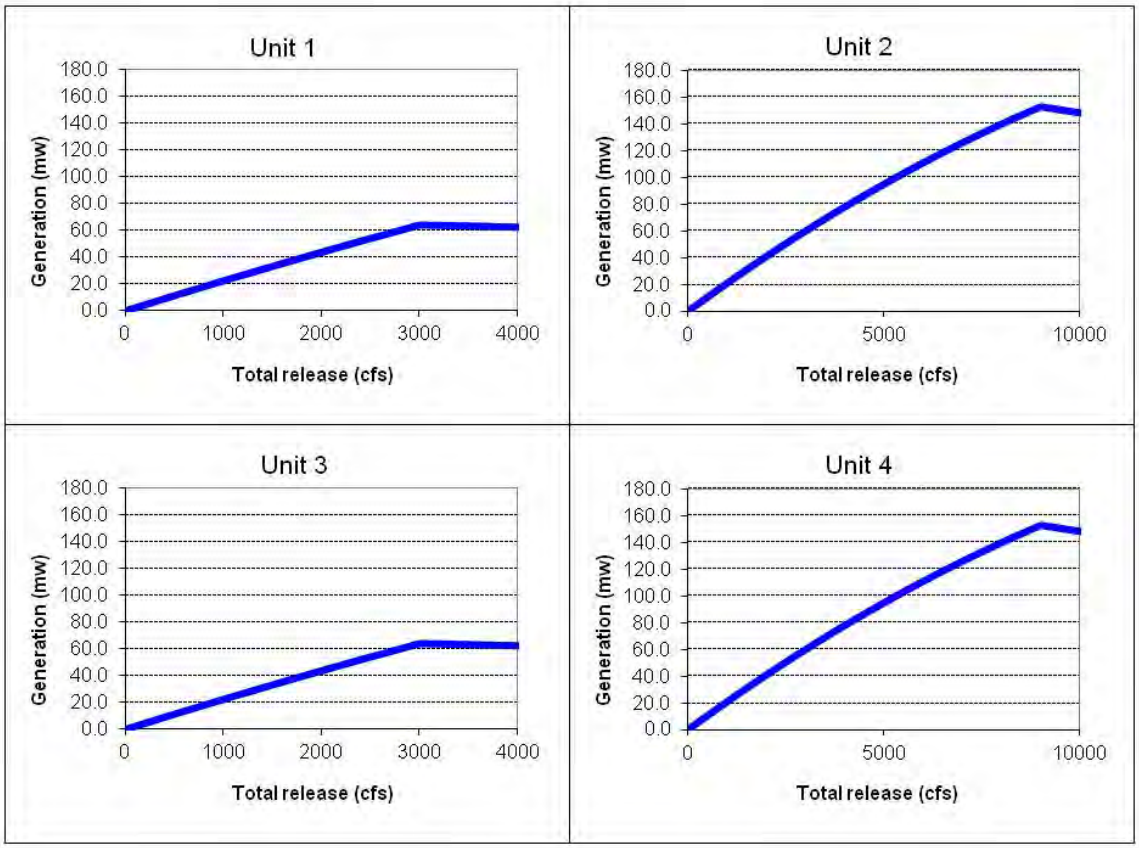

Figure 29.—Dispatch Problem 3 Release and Generation Characteristics.

Each of these generation units has minimum and maximum operational release capacities. For Units 1 and 3, the minimum release capacity is 20 cfs and the maximum release capacity is 3000 cfs. In Test Problem 3, neither Unit 1 nor Unit 2 have any prohibited operating zones. For Unit 2 and Unit 4 the minimum release capacity is 200 cfs and the maximum release capacity is 9000 cfs. In Test Problem 3, neither Unit 2 nor Unit 4 have any prohibited operating zones. The release constraints for all of the units in Test Problem 3 are summarized in Table 14.

Table 14.—Dispatch Test Problem 3 Characteristics

Unit	Release Constraints		Prohibited Zone 1		Prohibited Zone 2	
	MinQ (cfs)	MaxQ (cfs)	LowerQ (cfs)	UpperQ (cfs)	LowerQ (cfs)	UpperQ (cfs)
1	20	3000				
2	200	9000				
3	20	3000				
4	200	9000				

Dispatch Test Problem 4

Dispatch Test Problem 4 has four total generation units. These consist of two small generation units (Unit 1 and Unit 3), with a nominal capacity of 64 MW each, and two large generation units (Unit 2 and Unit 4), each with a nominal capacity of 153 MW. The relationship between release and generation for all of these units is shown in Figure 30.

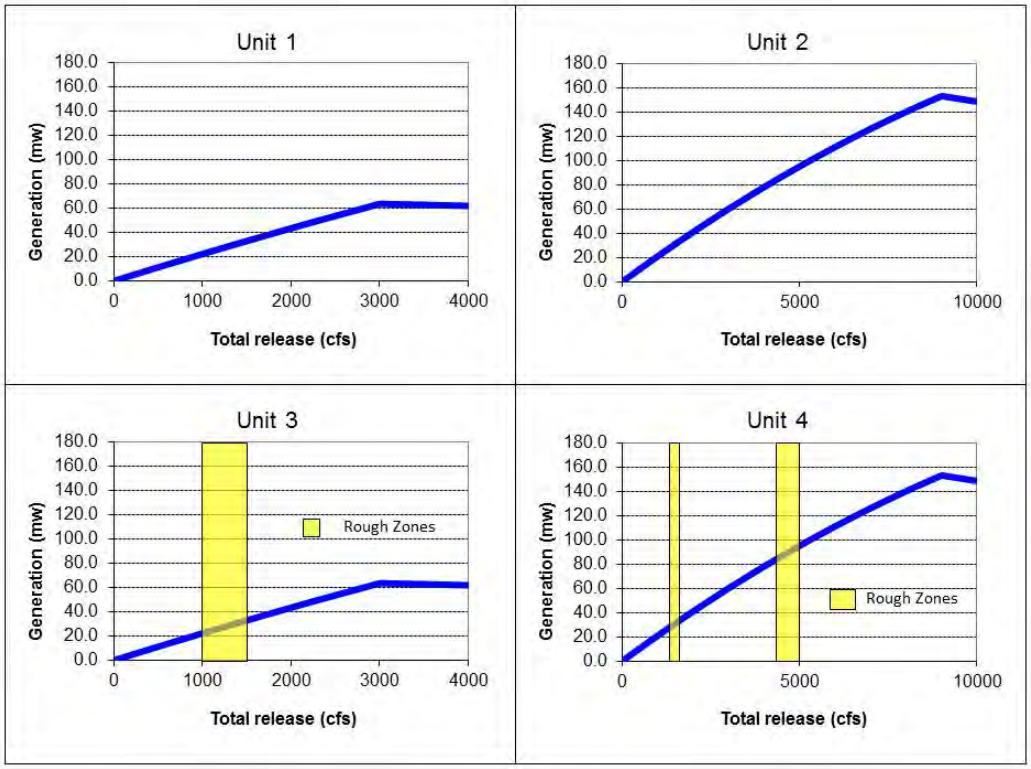

Figure 30.—Dispatch Problem 4 Release and Generation Characteristics.

Each of these generation units have minimum and maximum operational release capacities. For Unit 1 and Unit 3, the minimum release capacity is 20 cfs and the maximum release capacity is 3000 cfs. In Test Problem 4, Unit 3 has a single prohibited operating zone. The lower boundary for this prohibited operating (zone 1) is 1000 cfs. The upper boundary for prohibited operation zone 1 is 1500 cfs. For Units 2 and 4, the minimum release capacity is 200 cfs and the maximum release capacity is 9000 cfs. In Test Problem 4, Unit 4 has two prohibited operating zones. The lower boundary for prohibited operating zone 1 in Unit 4 is 1300 cfs. The upper boundary for this prohibited operation zone is 1700 cfs. For prohibited operating zone 2 Unit 4, the lower boundary is 4000 cfs. The upper boundary for this prohibited operation zone 2 is 5000 cfs. The release

constraints and prohibited operating zones for all of the units in Test Problem 4 are summarized in Table 15.

Table 15.—Dispatch Test Problem 4 Characteristics

Unit	Release Constraints		Prohibited Zone 1		Prohibited Zone 2	
	MinQ (cfs)	MaxQ (cfs)	LowerQ (cfs)	UpperQ (cfs)	LowerQ (cfs)	UpperQ (cfs)
1	20	3000				
2	200	9000				
3	20	3000	1000	1500		
4	200	9000	1300	1700	4000	5000

Appendix 7. Condensing, Motoring and Leakage

Unit Leakage

Depending on the plant, its design, maintenance condition and age, each turbine unit may unavoidably release some amount of water, even when it is not operating. This is termed, "leakage" and occurs to a greater or lesser extent in all hydropower plants. The major (but by no means only) source of leakage is passage of water around and through the closed wicket gates. When the unit is not operating, the water being released is wasted, since it does not drive the turbine runners and produces no electrical energy. A perfect seal around the wicket gates is not typically achievable and some level of leakage is typically expected, and tolerated.

Unit State and Leakage

For purposes of this modeling effort, each generator/turbine unit can be in any one of four states. These are (a) Dry maintenance/repair, (b) Off-line but operational, (c), Motoring/Condensing, and, (d) Operational. Table 16 summarizes the presence or absence of leakage compared across different unit states.

Table 16.—Unit State, Dispatch and Leakage

Unit State	Leakage (1=yes, 0=no)	Available for Dispatch (1=yes, 0=no)	Electric Energy Production (MW)
Dry Maintenance	0	0	0
Off-line	1	1	0
Motoring/Condensing	1	1*	negative (-)
Operational	0	1	positive (+)
*If under automatic generation control (AGC) and depending on the applicable market rules.			

When a unit is under dry maintenance or repair, the unit is dewatered, the wicket gates are isolated and leakage is minimized. If there is any leakage while the unit is in this state, it is generally reduced to insignificant levels. Some degree of leakage is expected to occur when the turbine/generator units are off-line and also

when they are being used in condensing mode. When a unit is in the operating or generating state, all of the water released by the unit is being used to drive the turbine runners and, by definition, no leakage occurs. In this state, model leakage is assumed to be zero.

The modeling implications of Table 16 are fairly straightforward. First, if a unit is in the dry maintenance/repair state, it is unavailable for dispatch and it is assumed not to leak. From a modeling standpoint, this unit cannot be dispatched and will not appear in the available unit list. Second, if a unit is available for dispatch, but is not currently in operation, it is leaking. From a modeling standpoint, there are some unavoidable water releases from all nonoperational units. These collateral releases are not subject to modeling control but should be accounted for in the calculation of aggregate plant release. Interestingly enough however, a nonscientific survey of Reclamation power dispatchers suggests this source of leakage is not explicitly accounted for in management decisions or in the dispatch decision. A further modeling difficulty is that leakage from this source cannot be known, before the dispatch decision is made. For these reasons, the unit dispatch model does not consider potential leakage from operational but idle units. The designation of a unit for condensing/motoring is typically an operator based decision, outside of modeling control. Selection of a unit for condensing removes that unit from the list of units available for dispatch. The leakage which results from condensing unit(s), while not subject to model control, is accounted for in the calculation of total plant release. Finally, if a unit is operational (has been dispatched), all water released from that unit is used to produce electrical energy and no leakage occurs. The releases from operational units are subject to full modeling control and leakage is assumed to be zero for those units.

Characterizing Leakage in the Model

Leakage from any particular hydropower plant varies considerably depending on the design, installed equipment, its maintenance condition and age. In addition, it may vary with the elevation of the forebay reservoir (hence, the head). The data on leakage at Reclamation hydropower plants is limited but initial indications suggested it might be quite substantial. For seven plants, Steves (personal communication, 2012) calculated that leakage at zero generation output is in the range of about 16-percent of the total turbine release capacity, measured in cubic feet per second (cfs). Subsequent explorations of this problem with other Reclamation Staff provided more definitive evidence that leakage at zero generation was much lower, on the order of 0.5 to 2 percent of the total turbine release capacity.

With this additional information as guidance, a leakage of 1.0-percent of the total unit release capacity for all units which are not operating, but are available for dispatch, is used as the default value in the unit dispatch model. To accommodate

the effects of greater or lesser degrees of unit leakage, the user may vary the amount of leakage in any given model run.

Unit Condensing and Motoring

Depending on the design of the plant, hydro units can be operated as synchronous condensers to increase reactive power on the interconnected grid. In this mode (also referred to as "motoring"), the hydro unit's breaker switch is closed and no water is released to drive the turbine. Air is pumped into the turbine to allow it to spin freely. Real electric power is supplied to the unit and the unit's generator acts like a motor, rotating the turbine shaft and supplying reactive power to the system. Depending on the control settings, condensing units can also be used for voltage and frequency control in the system.

Characterizing Condensing in the Model

The designation of a unit for condensing/motoring is commonly an explicit decision made by the plant operator. This decision is outside of modeling control. Selection of a unit for condensing removes that unit from the list of units available for dispatch. A condensing unit has two important features that should be characterized by the model. First, a condensing unit requires a supply of real electric energy to turn the generator. This is summarized in Table 16. For modeling purposes, the amount of real energy required for condensing is assumed to be 5% of the generator's rated maximum output capacity, measured in MW. It is assumed the other operating generation units must produce this additional energy, in addition to their otherwise planned output levels, to support unit condensing operations. To accommodate the effects of greater or lesser real power requirements to enable unit condensing, the user may vary the amount of condensing energy required in any given model run. A second feature which should be considered is that a condensing unit is subject to leakage. A leakage of 16-percent of the total unit release capacity for all condensing units is used as the default value in the unit dispatch model. To accommodate the effects of greater or lesser degrees of unit leakage, the user may vary the amount of leakage in any given model run. The leakage which results from condensing unit(s), while not subject to model control, must be (and is) accounted for in the calculation of total plant release.

Appendix 8. Triangular Rough Zone Penalty

Constraints separate the solution space into feasible and infeasible spaces. The resulting feasible solution space is generally limited and can be discontinuous, even when the optimization problem is not. Constraint equations can be linear or nonlinear in nature. In general, there are three classes of constraint equations. These broad classes are; boundary constraints, equality constraints and inequality constraints.

Research on the incorporation of constraints in evolutionary programming methods and the solution of constrained optimization problems with evolutionary methods is rather voluminous. Carlos Coello Coello published a widely cited synopsis of this work (Coello Coello 2002) and maintains an online annotated bibliography summarizing this ever-expanding body of research (http://www.cs.cinvestav.mx/~constraint/index.html). As of February 2012, this bibliography exceeded 115 pages in length.

The Phase 1 Report (Harpman 2012) reviewed the complete spectrum of constraint handling technologies employed in traditional (calculus based) and evolutionary algorithms. One of the commonly employed constraint handling methods is to penalize or mathematically disadvantage the fitness value for solutions which are infeasible. This approach is widely used both in traditional calculus based applications as well as in the application of evolutionary algorithms. In the case of a maximization problem, fitness (F) is typically defined as the sum of the objective function value (f(x) *minus* the infeasibility penalties (P), if any. This is shown in equation (33).

(33) $$F=f(x) - P.$$

The unit dispatch problem examined in this Phase 2 research effort has some of the same types of constraints examined in the Phase 1 effort including boundary constraints, inequality constraints and equality constraints. In addition, the unit dispatch problems examined here have rough zone constraints, also known as prohibited operating zones.

Mathematically speaking, the presence of rough zones creates a piece-wise and discontinuous problem, a class of optimization problems which is much more difficult to solve. Figure 31 conceptually illustrates a unit dispatch problem with two rough zones, Rough Zone 1 and Rough Zone 2. These two rough zones divide the flow versus generation function into three discrete sections; one

segment is from zero to the lower limit of Rough Zone 1, the next segment is from the upper limit of Rough Zone 1 to the lower limit of Rough Zone 2 and the final segment is from the upper limit of Rough Zone 2 onward.

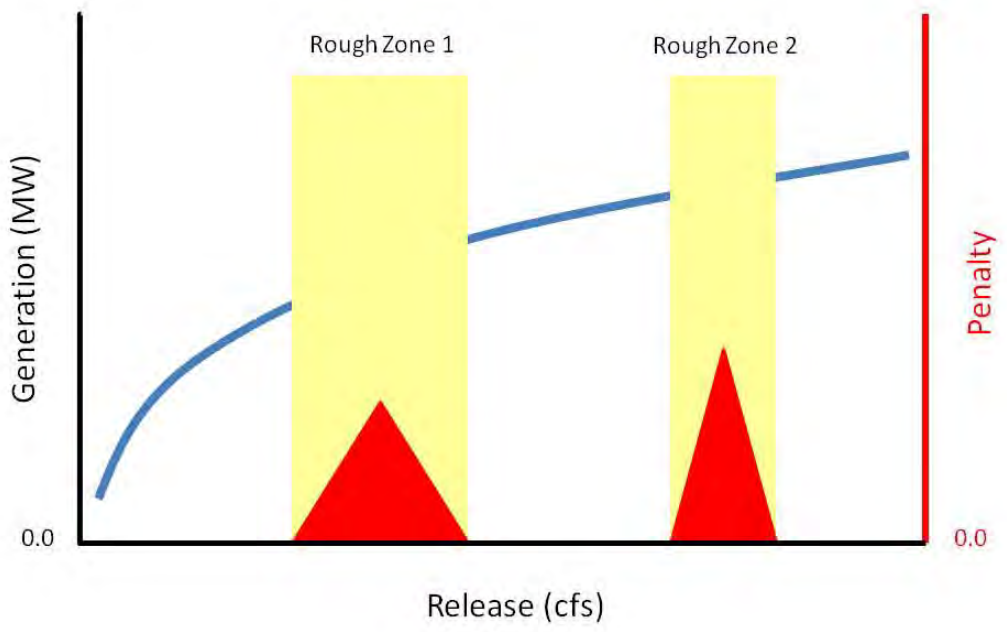

Figure 31.—Rough Zones and Penalties.

Mechanically and operationally, the generation unit being dispatched should be operated within one of the permissible ranges, and not within one of the rough zones. The generation unit should be operated below the lower limit of Rough Zone 1, or in the range between the upper limit of Rough Zone 1 and the lower limit of Rough Zone 2, or somewhere in the region above the upper limit of Rough Zone 2.

As part of this research effort, a triangular penalty function was developed to characterize rough zones and efficiently avoid the dispatch of generation units within them. These triangular rough zone penalty functions mathematically disadvantage solutions falling within a prohibited operation zone but are zero for solutions within permissible operation regions. A non-zero penalty is assessed any solution greater than the lower limit of the rough zone or less than the upper limit of the rough zone. As the name implies, the numerical magnitude of the penalty is triangular. The penalty linearly increases as a potential solution moves from the lower limit of the rough zone towards the middle of the rough zone and reaches a maximum half-way between the lower limit and upper limit of the rough zone (the apex of the triangle). The magnitude of the penalty then diminishes linearly as the solution moves from the center of the rough zone towards the upper limit of the rough zone.

Figure 31 illustrates the triangular rough zone penalty function. To understand this concept more fully, it is helpful to take note of the common horizontal axis (release (cfs)) and note the right-hand vertical axis indicates the absolute value of the penalty. As shown in Figure 31 the penalty is denoted in red and the numerical value of the penalty (P) is zero for releases outside of the rough zones. The penalty increases within a given rough zone, reaches a maximum at the center of the rough zone and then decreases as the potential release approaches the upper limit of a rough zone.

The triangular rough zone penalty approach efficiently conveys information about the feasibility status of a solution to the evolutionary solution algorithms. Potential solutions outside of the rough zone are preferred to solutions within the rough zone and infeasible solutions are not uniformly penalized as might be the case with less sophisticated techniques. Potential solutions near the outside limits of the rough zone are preferred (or are less penalized) over solutions at the middle of the rough zone (which are penalized more).

The triangular rough zone penalty function approach has proven to be quite effective for the unit dispatch test problems examined. This approach serves to efficiently disadvantage the set of solutions which are infeasible, relative to those which are feasible. In the evolutionary algorithm context, the individuals with the greatest fitness form the basis for potential solutions in succeeding generations. As a result, the population will select for, or move towards, the feasible solution space, with each successive iteration.

Appendix 9. Box and Whisker Plots

Box and whisker plots provide a quick summary of important dataset characteristics including the central tendency, dispersion, asymmetry, and extreme values. These plots are based on descriptive statistics of the underlying empirical distribution and are nonparametric or distribution-free. Consequently, they do not reflect any of the assumptions associated with distributions, such as the normal distribution. They provide an effective way of identifying asymmetrical attributes in datasets. Perhaps most importantly, the graphically compact nature of box and whisker plots facilitates rapid side-by-side comparison of different samples or datasets, which can otherwise be difficult to interpret.

Box and whisker plots were first proposed by statistician John Tukey circa 1970. Their application has become relatively commonplace and standardized. Consumers of technical literature recognize there are some variations in the way these plots are defined. As related by Banacos (2011) the majority of these variations are in the definition of the "whiskers."

For purposes of this research project, the 5th and 95th percentiles are used in this document for the ends of the whiskers. Using this convention, there is a 5% probability a data point will fall beyond the high or low values at the ends of the whiskers. The range between the whiskers encompasses 90% of the empirical distribution.

Figure 32 illustrates a representative Box and Whiskers plot as employed in this study. As shown in Figure 32, the median (black horizontal line within the box) illustrates the central tendency measure of the plotted data set. The median is the 50th percentile point, or the point where one-half the data lie above and one-half the data lie below. The "box" illustrates the inter-quartile space between the 75th percentile and the 25th percentile of the empirical data distribution. The size of the box provides an indication of how much variance or dispersion are reflected in the data. The median point may or may not be in the center of the box depending on whether the data is symmetric around the median or is asymmetric (skewed). In Figure 32 the whiskers extend to the 95th percentile and the 5th percentile of the empirical distribution. The dispersion of the empirical distributions are indicated by the distance between the ends of these whiskers—the greater the distance between them, the more the dispersion and the shorter the distance between the ends of the whiskers, the less the dispersion. If the empirical distribution is symmetric, the whiskers will be the same length. If the distribution is asymmetric, the upper and lower whiskers will be of differing lengths.

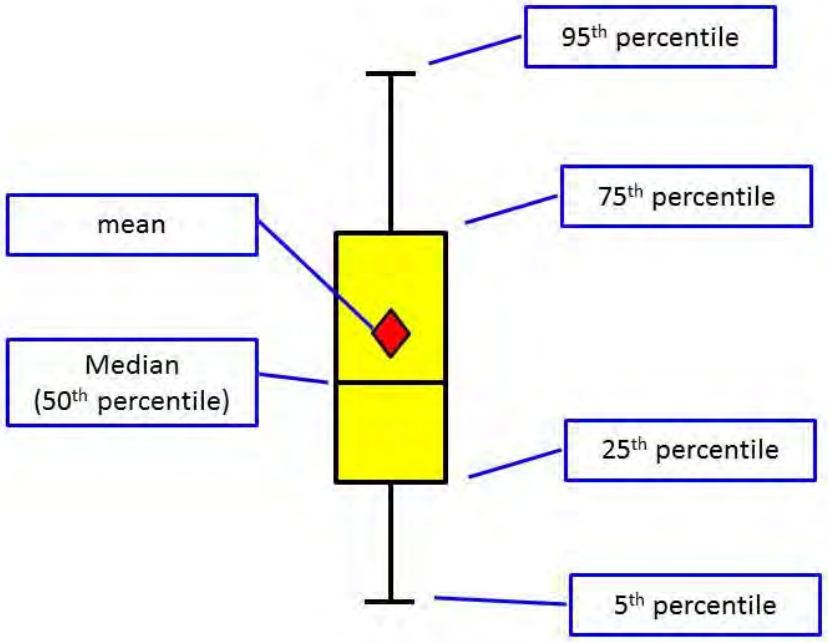

Figure 32.—Interpreting a Box and Whisker Plot.

There is no built-in functionality for generating a Box and Whisker plot using Microsoft Excel. However, using the tedious and lengthy set of steps described in the very helpful Peltier Tech Blog (www.peltiertech.com), Excel can be used to create these helpful summaries. These box and whisker plots illustrate the characteristics of empirical distributions and facilitate the comparison of different samples.

Appendix 10. Initialization Experiments

The numerical results of replicated initialization experiments (ntrials=50) using different initialization approaches for dispatch test problem 4 with a release of 10,000 cfs are reported in this appendix. Except where noted in the results tables, all of the algorithm parameters were set to their default values, as described in the text. Some of these results are summarized in graphic form in the main body of the report.

Table 17.—DDE Iterations to Convergence by Initialization Type.

	Uniform	Neider	Weyl/Torus	Haber	Halton	OBL
nobs	50	50	50	50	50	50
mean	289.1	285.4	298.6	283.3	281.4	291.5
std. dev	37.7	29.9	35.5	38.7	36.1	38.5
minimum	210.0	228.0	242.0	214.0	222.0	201.0
05th perc	236.8	247.9	247.5	232.9	239.7	243.8
25th perc	262.0	262.5	273.3	261.3	259.5	262.3
median	283.0	286.0	298.5	279.0	278.5	288.0
75th perc	313.5	308.8	324.8	305.0	289.0	315.8
95th perc	355.5	332.0	355.8	360.7	368.0	364.0
maximum	381.0	368.0	380.0	395.0	395.0	392.0

Table 18.—PSO Iterations to Convergence by Initialization Type.

	Uniform	Neider	Weyl/Torus	Haber	Halton	OBL
nobs	50	50	50	50	50	50
mean	94.1	92.8	81.7	79.5	95.1	87.2
std. dev	59.1	76.3	27.8	26.9	48.8	40.0
minimum	32.0	36.0	32.0	35.0	31.0	33.0
05th perc	34.5	46.8	48.5	49.0	47.5	35.0
25th perc	58.5	62.3	60.0	60.8	63.3	62.5
median	72.0	77.0	74.5	72.5	85.0	82.5
75th perc	112.5	99.0	98.8	97.5	110.8	99.8
95th perc	205.8	141.5	127.0	120.1	191.2	159.1
maximum	311.0	581.0	168.0	165.0	307.0	221.0

Table 19.—RCGA Iterations to Convergence by Initialization Type.

	Uniform	Neider	Weyl/Torus	Haber	Halton	OBL
nobs	50	50	50	50	50	50
mean	2529.0	2493.9	2400.0	2445.6	2613.4	2250.9
std. dev	1263.7	1392.0	1741.5	1550.2	1362.6	1277.5
minimum	597.0	147.0	214.0	237.0	98.0	127.0
05th perc	857.9	548.4	629.7	702.1	496.1	476.1
25th perc	1562.8	1493.8	1299.3	1353.3	1687.8	1229.0
median	2316.5	2191.0	1790.0	2075.5	2525.0	2207.5
75th perc	3120.0	3454.8	3063.3	3115.8	3631.8	2830.5
95th perc	4755.0	5065.0	6206.5	5538.6	5010.9	4866.5
maximum	5918.0	6012.0	8100.0	7209.0	5547.0	5587.0

Appendix 11. Stopping Rule Experiments

The numerical results of replicated stopping rule experiments (ntrials=50) using different stopping rules for dispatch test problem 4 with a release of 10,000 cfs are reported in this appendix. Except where noted in the results tables, all of the algorithm parameters were set to their default values, as described in the text. Some of these results are summarized in graphic form in the main body of the report.

The table and the figure below detail the results obtained with the PSO algorithm using each of the stopping rules. As shown, the intelligent rules (Pop_SD, Elite_SD, and Elite_Mean) greatly decrease the computational time and expense required to achieve convergence on dispatch test problem 4. The Elite_Mean approach appears to have a small computational advantage relative to the other approaches. However a formal statistical investigation of this possibility remains to be undertaken.

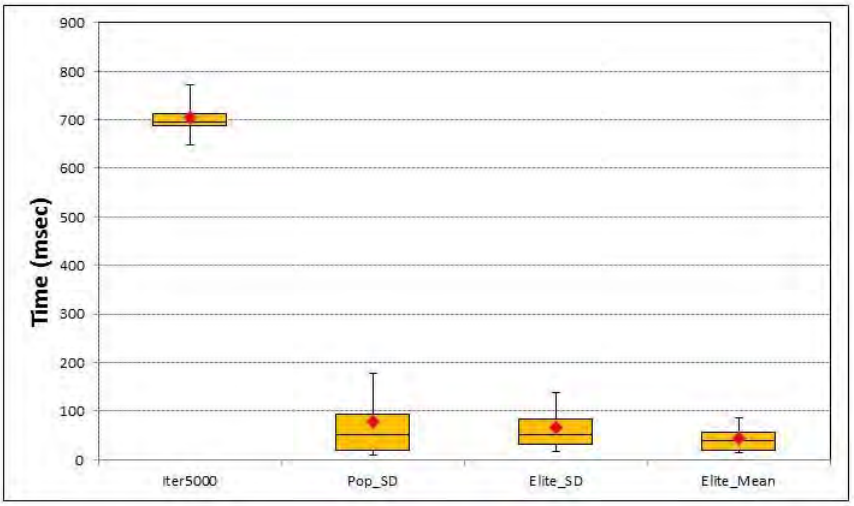

Figure 33.—PSO Stopping Rule Results.

Table 20.—PSO Numerical Results of Stopping Rule Experiments.

	Iter5000	Pop_SD	Elite_SD	Elite_Mean
nobs	50	50	50	50
mean	704.3	77.6	67.1	43.0
std. dev	35.2	119.4	71.2	27.5
minimum	615.0	9.0	14.0	13.0
05th perc	649.4	10.5	16.4	14.0
25th perc	687.3	18.5	31.5	20.3
median	695.0	51.5	50.5	40.0
75th perc	713.3	94.8	82.5	55.8
95th perc	772.9	177.5	137.9	87.3
maximum	786.0	801.0	503.0	146.0

Experiments comparing the population standard deviation stopping rule to the elite standard deviation were also undertaken. Figure 34 illustrates the results of those experiments. Although the elite approach might be expected to reduce convergence time, this expectation does not hold across all of the algorithms. Potentially, this counterintuitive result stems from the additional time required to sort the observations to identify the elite fraction.

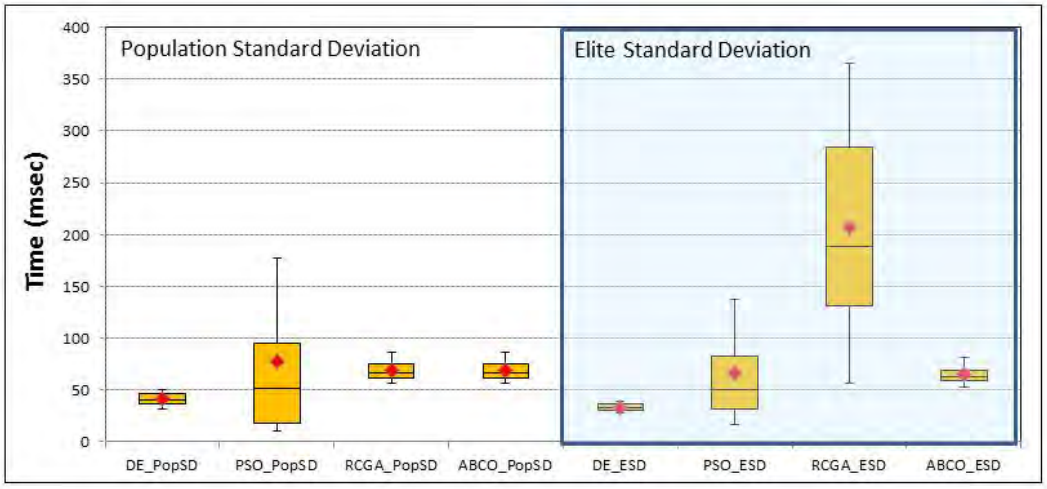

Figure 34.—Elite and Population SD Stopping Rule Results.

Appendix 12. Problem Size Experiments

The numerical results of replicated experiments (ntrials=50) comparing the solution time for different sized unit dispatch problems are reported in this appendix. Some of these results are summarized in graphic form in the main body of the report.

The values in these tables are CPU time measured in milliseconds (msec).

Table 21.—Problem 1 (2-Unit) Convergence Results.

	DE_P1	PSO_P1	RCGA_P1	ABCO_P1
nobs	50	50	50	50
mean	12.3	18.9	167.3	14.0
std. dev	1.6	8.2	78.7	0.8
minimum	9.0	9.0	28.0	12.0
05th perc	10.0	11.0	46.0	13.0
25th perc	11.0	13.0	118.0	14.0
median	12.0	16.0	158.0	14.0
75th perc	13.0	23.0	216.8	14.8
95th perc	15.0	37.2	307.0	15.0
maximum	17.0	44.0	338.0	15.0

Table 22.—Problem 3 (4-Unit) Convergence Results.

	DE_P3	PSO_P3	RCGA_P3	ABCO_P3
nobs	50	50	50	50
mean	33.5	26.3	188.7	65.4
std. dev	4.1	12.8	112.9	9.4
minimum	26.0	9.0	27.0	52.0
05th perc	26.5	10.0	45.2	53.0
25th perc	31.3	17.0	113.5	59.0
median	33.0	21.5	163.5	65.0
75th perc	36.0	37.0	231.3	68.8
95th perc	41.0	49.7	398.3	81.1
maximum	43.0	55.0	535.0	99.0

Appendix 13. Rough Zone Experiments.

The numerical results of replicated experiments (ntrials=50) comparing the solution time for problem 1 (2-units, no rough zones) and problem 2 (2-units, with rough zones) are reported in this appendix. The construction of this problem with a release level of 7500 cfs ensures the rough zone constraint is binding in problem 2. Some of these results are summarized in graphic form in the main body of the report.

The values reported in these tables are the CPU time measured in milliseconds (msec).

Table 23.—Problem 1 Convergence Results (Q=7500).

	DE_Q75	PSO_Q75	RCGA_Q75	ABCO_Q75
nobs	50	50	50	50
mean	12.7	16.0	145.1	14.4
std. dev	1.4	2.7	69.2	0.9
minimum	8.0	12.0	29.0	12.0
05th perc	11.0	13.0	39.8	13.0
25th perc	12.0	14.0	91.5	14.0
median	13.0	15.0	139.5	14.5
75th perc	13.0	18.0	200.8	15.0
95th perc	15.0	20.6	262.4	15.0
maximum	16.0	22.0	284.0	16.0

Table 24.—Problem 2 Convergence Results (Q=7500).

	DE_Q75	PSO_Q75	RCGA_Q75	ABCO_Q75
nobs	50	50	50	50
mean	21.3	37.7	69.1	17.8
std. dev	2.1	8.2	51.4	9.3
minimum	18.0	21.0	6.0	12.0
05th perc	18.0	28.0	7.0	12.0
25th perc	20.0	34.0	34.3	13.0
median	21.0	37.0	54.5	14.0
75th perc	22.8	40.0	102.0	18.0
95th perc	25.0	47.0	180.6	36.6
maximum	27.0	78.0	214.0	58.0

Appendix 14. PSO Parameter Experiments

The numerical results of replicated experiments (ntrials=50) comparing the solution time for different values of the social weight (c2) parameter used in the PSO algorithm are reported in this appendix. Some of these results are summarized in graphic form in the main body of the report.

The values in these tables are the number of iterations to convergence.

Table 25.—Results of C2 Parameter Experiments.

	c2=1.3	c2=1.5	c2=1.7	c2=1.9	c2=2.1	c2=2.3
nobs	50	50	50	50	50	50
mean	1663.1	176.9	115.0	93.0	79.7	86.8
std. dev	1840.9	157.3	146.8	51.1	35.2	41.1
minimum	85.0	37.0	31.0	31.0	26.0	28.0
05th perc	94.8	38.5	36.0	46.4	40.3	40.0
25th perc	229.3	72.3	63.3	62.0	57.3	61.0
median	751.5	141.0	83.0	80.5	72.0	76.0
75th perc	2386.8	235.3	119.0	103.5	92.3	99.5
95th perc	5028.4	366.6	196.4	204.5	146.9	177.1
maximum	7283.0	972.0	1032.0	270.0	215.0	218.0

Appendix 15. DE Mutation Strategies

The numerical results of replicated experiments (ntrials=50) comparing the solution time for different mutation strategies used in the DE algorithm are reported in this appendix. Some of these results are summarized in graphic form in the main body of the report.

The values in these tables are the number of iterations to convergence.

Table 26.—DE Mutation Strategy Experiments.

	Rand1	Best1	RandSF	Trigon
nobs	50	50	50	50
mean	150.0	497.2	509.4	3147.6
std. dev	18.8	105.9	114.8	1011.6
minimum	101.0	262.0	268.0	1961.0
05th perc	120.5	340.7	327.1	2054.5
25th perc	138.0	429.0	416.3	2499.0
median	150.5	484.5	523.0	2869.5
75th perc	166.0	561.3	590.0	3554.8
95th perc	177.0	668.6	690.1	5425.6
maximum	180.0	771.0	756.0	6331.0

Table 27.—DE Mutation Experiments (continued).

	Detvsf	SelfAdapt	Best2	Rand2
nobs	50	50	50	50
mean	1077.7	287.1	1162.7	1772.1
std. dev	194.8	34.6	327.0	394.6
minimum	672.0	203.0	603.0	873.0
05th perc	748.0	249.9	771.2	1142.7
25th perc	957.5	263.0	901.3	1503.0
median	1086.0	284.5	1105.5	1808.0
75th perc	1194.8	305.8	1353.3	2046.5
95th perc	1361.9	353.3	1758.2	2442.2
maximum	1615.0	381.0	2090.0	2591.0

Appendix 16. Program Dictionary

The table below contains the names of the programs used in this analysis and a description of their purpose. This will help facilitate location of the source code files and their reuse at a later date.

Table 28.—Program Dictionary

EA	Filename	Purpose
DE	de04	Development
DE	uddeXE2	Economic dispatch
DE	uddeXE2prod	Testing environment
PSO	pso4	Development
PSO	udpsoXE2	Economic dispatch
PSO	udpsoXE2prod	Testing environment
RCGA	rcgen	Development
RCGA	udrcgaXE2	Economic dispatch
RCGA	udrcgaXE2prod	Testing environment
ABCO	abc04_xe2	Development
ABCO	udabco_xe2	Economic dispatch
ABCO	udabcoxe2prod	Testing environment

Mission Statements

The U.S. Department of the Interior protects America's natural resources and heritage, honors our cultures and tribal communities, and supplies the energy to power our future.

The mission of the Bureau of Reclamation is to manage, develop, and protect water and related resources in an environmentally and economically sound manner in the interest of the American public.