Simulated Annealing Articles summaries

William Bezuidenhout

Tuesday, 11 June 2010

1 NAIS: A Calibrated Immune Inspired Algorithm to Solve Binary Constraint Satisfaction Problems

```
@inproceedings{ZunigaRM07,
  title = {NAIS: A Calibrated Immune Inspired Algorithm to Solve Binary Constraint
  author = {Marcos Zuniga and Mara-Cristina Riff and Elizabeth Montero},
  year = {2007},
  doi = {http://dx.doi.org/10.1007/978-3-540-73922-7_3},
  pages = {25-34},
  booktitle = {Artificial Immune Systems, 6th International Conference, ICARIS 200
  editor = {Leandro Nunes de Castro and Fernando J. Von Zuben and Helder Knidel},
  volume = {4628},
  series = {Lecture Notes in Computer Science},
  publisher = {Springer},
  isbn = {978-3-540-73921-0},
}
```

Immune artificial systems as well as evolutionary algorithms are very sensitive to the values of their parameters.

2 Accelerating genetic algorithm computation in tree shaped parallel computer

```
@article{245894,
 author = {H\"{a}m\"{a}l\"{a}inen, Timo and Klapuri, Harri and Saarinen, Jukka and
 title = {Accelerating genetic algorithm computation in tree shaped parallel compu
 journal = {J. Syst. Archit.},
```

```
volume = {42},
number = {1},
year = {1996},
issn = {1383-7621},
pages = {19--36},
doi = {http://dx.doi.org/10.1016/1383-7621(96)00009-4},
publisher = {Elsevier North-Holland, Inc.},
address = {New York, NY, USA},
}
```

Genetic Algorithms are one ofthe most interesting novel methods for reducing the search effort and time in optimization tasks. The basically simple concept of artificial evoltion is very flexible and can be aplpied to a variety of problems, including those that are hard to model mathematically. However, applications may represent very different problem domains and therefore the style of GA may differ significantly from one application to another. In general GAs are strongly applicaiton dependent, which results in an abundance of their variations and extensions. In addition to the the requirements of the problem at ahnd, it is important for their usage how to enhance the quality of the search and how to reduce the search time. Some improvoement can be obtained by adjusting the algorithm and softare, but for example futher reduction in the search time requires also improvements in the computing hardware.

Typical GA flow diagram on page 20

2.1 Sequantial Genetic Algorithm

In the first pahase the problem is encoded to an individual. The actual computation starts in the next phase, in which a number of randomly generated individuals are created to form an initial population. After that, the fitness of each individual is evaluated and if a sufficient is found, the GA execution is terminated. Otherwise, some method is used to select parents among individuals for the reproduction phase. In this phase a new generation of the population is created or the old population is updated, and previous phases are repeated.

The loop of fitness evaluation, reproduction and replication may be performed a number of times before a satisfactory solution has been found. The more time one phase consumes, the larger will be the total search time. On the other hand, the larger the population and the longer the individual strings, the more time each pahse tkes. Although the more time needed

for each phase is dependent on the problem as well as computing platform, some general estimation can be given. The initialization phase is needed only once at the beginning of the execution and thus it has minor contribution to the total search time. In contrast, the fitness evaluation may be very complicated, so that it consumes most of the total execution time. The reproduciton phase consists of selection of mates and string manipulations. The selection scheme may be complication and requires many random numbers , for which reason this phase may also be timeconsuming. String manipulations take least time, especially when strings are binary valued.

In order to reduce tthe total search time one has to try to imporve the search quality and speed up the execution of single phases. The search quality can be adjusted by parameters like mutation rate and crossover probability.

3 A combined branch-and-bound and genetic algorithm based approach for a flowshop scheduling problem

```
Journal Title  - Annals of Operations Research
    Article Title  - A combined branch-and-bound and genetic algorithm based appro
    Volume  - Volume 63
    Issue  - 3
    First Page  - 397
    Last Page  - 414
    Issue Cover Date  - 1996-06-28
    Author  - Amit Nagar
    Author  - Sunderesh S. Heragu
    Author  - Jorge Haddock
    DOI  - 10.1007/BF02125405
    Link  - http://www.springerlink.com/content/TQH4417L16381R67
```

Using information concerning the solution space's regions and guided by two operators - reproduction and crossover - a GA searches for optimal or near-optimal solutions.

The reproduction operator samples the individual schedules in proportion to their objecive function values (OFVs). A schedule with an OFV higher than the population average hasa greater propbaility of being sampled than scheudles iwth lower OFVs(for a problem with maximization objective). By explicitly testing the goodness of a single schedule, is is clear

that GA implicitly tests a number of different regions determined by the job positions in the schedule.

Reproduction alone cannot guide Ga to the optimal solution. IN the absence of other operators, it will simply give us the best schedule from the initial populations iwthout exploring other regions of the solutions space. To provide diversity of GA's search, the crossover operator is used. It is a matching and swapping operator which combines the partial solutions from two "parent" schedules to produce two new "offspring" schedules. Crossover perfoms two functions:

- samples new schedules from the target area (exploitation), and
- directs the search into new regions of the solutions spaces (Exploration).

It is the role of the crossover operator to provide more samples of good regions from the subset of schedules defined by each. This will enable GA to evaluate the target regions based on a large number of samples. Thus, the final evaluation of the goodness of the regions has a sound basic and is not chance occurence. In its exploratory phase, GA destroys certain partial solutions and forms new partial ones. This allows it to guide its search into new regions of the solutions space. Obtaining a balance between exploration and exploitation is a fundamental problem in the theory of adaptation.

Thus, it is a combination of exploration and exploitation that allows Fa to sample already targeted regions to confirm or disprove the previous estimates while exploring new regions in the solution space. GA attempts to ensure that there is an optimal allocation of jobs to schedule positios. This in turn improves the possibility of obtaining a near optimal schedule at the end of a run.

It should be noted that the efficiency of GA in the meta-heuristic depends upon the information made available by the branch-and-bound algorithim, which itself depends upon the node level to which it is allowed to proceed. Thus, an increase in the GA's efficiency comes at the expense of more computational effort by the branch-and-bound algorithm.

To obtain the initial population, most GA's randomly rengerate the required number of solutions (schedules). However, in our implementaiton, we generate a large part of the initial population using the heuristic that provides us with the upper bound in the brand-and-bound algorithm. To do so, we first obtain a schedule using the heuristic. We then explore the neighborhood of this schedule and retain those with an OFV within a certain percentage of the heuristic schedules's OFV. (The neighboring schedules are

generated by performing parwise and 3-opt job interchanges) If the initial population consists entirely of schedules generated using a local minimum algorithm, is is likely that the GA search will converge prematurely, prior to finding the global optimum. To avoid such a occurrence and maintain enough diversity in the population, we generate a percentage of the population randomly.

In the traditional or genereic GA, a pair of selected schedules is used by the crossover operators to generate offsprings. Intuitively, it appears that if we generate offsprings selectively from "good" parents, the quality of the resulting offsprings is likely to be superior.

If a each selected schedule is in a suboptimal region, we then distrupt job positios in this schedule by means of a distruption operator. This distruption operator is termed as a static schedule distruption operator (SSD). The distruption is carried out by interchaning two jobs in the schedule.

Once the known sub-optimal schedules have been disrupted, the GA uses a crossover operator to produce offspring from parents. An important requirement of the crossover operator for ordering problems is to maintain feasibility. While a number of crossover operators such as cycle, order, etc., are available, we use the partially matched crossover (PMX) operator. It is a matching and swapping operator which perfroms these operations on a pair of selected schedules. PMX is a 2-point crossover operator. It works by selecting two points within a schedule. The jobs within the points are swapped (invariably leading to two infeasible schedules). A mapping operation is then used to remove this infeasibility from each of the resulting schedules.

Upon completion of a generation, reproduciton is used to generate a new population for the next generation. This is done by selecting schedules in proportion to their OFVs. A number of ways have been suggested in the literature for implementation of this operator. We use the RWS procedure for reproduction. Also, GA's are designed to maximize the objective function.

Conceptually, the meta-heuristic differs from the conventional GA in its interface with the branch-and-bound algorithm. The meta-heuristic uses a schedule distruption operator to efficiently guide its search by using the available information. The remaining operators for the meta-heuristic are similar to the ones in a conventional GA. We now discuss two modifications to the meta-heuristic.

3.1 MODIFICATION 1

In developing the schdule disruption operator for the meta-heuristic, we relied on static worst-case information about the objective function, i.e. information collected at he beginning of the run (hence the name static schedule disruption operator). This was then used by the embedded GA to disrupt known sub-optimal schedules. This seems to be a reasonable appraoch, since we are direction the GA towards certain "desireable" branches of the tree.

The Dynamic schedule disruption operator is dynamic in the sense that is uses current schedule information rather than information colected at the beginning of the run (as is done by SSD operator). Thus the DSD operator attempts to disrupt the current (sub-optimal) schedule and find an improved one, if possible. This operator is used in modification 1 instead of the SSD operator usedin the meta-heuristic.

3.2 MODIFICATION 2

The meta-heuristic works well when the embedded branch-and-bound algorithm can provide sufficient information to direect the GA from known sub-optimal regions. However, in large problems or those in which the processing time data are such that there is much idle time even in optimal/near optimal schedules, the branch-and-bound algorithm cannot provide much node fathing information since the gap between lower and upper bounds is large. As a result, it cannot provide sifficient problem-specific information to the embedded GA. When the average dominance values are close to zero, the problem is difficult for branch-and-bound. Average dominance is defined as the sum of processing tmes of all jobs on the first machine minus that on the second machine. Thus, for such problems, the meta-heuristic is likely to perform poorly. Hence, we have developed another modifed version of the meta-heuristic which does not use any information from the branch-and-bound procedure. Instead, modification2 uses the worse half of the population for selection the schedules for disruption.

In summary, modification 1uses the branch-and-bound algorithm to obtain node fathomg information but employes the DSD operator to disrupt sub-optimal schedules. On the other hand, modification 2 does not use the branch-and-bound algorithm, but uses the DSD operator to disrupt schedules that lie in the worst half of the population.

3.3 Results

Note that modification 1 is applied to problems in which the average dominance is not close to zero, i.e. those problems for which the branch-and-bound algorihtm provides abundant information for the embedded GA. This enables us to accurately compare the effect of the static and dunamic operators. Hhowever, based on the results for the meta-heuristic and modification 1, no conclusions can be made about the effectiveness for the two operators. Although intuitively the DSD operators appears to be superior to SSD, the computational results do not demonstrate this. The meta-heuristic and modification 1 out-perform each otehr over different data sets. As a result, modification 1 is excluded for larger problems.

4 Adaptive simulated annealing genetic algorithm for system identification

```
@article{Jeong1996523,
title = "Adaptive simulated annealing genetic algorithm for system identification"
journal = "Engineering Applications of Artificial Intelligence",
volume = "9",
number = "5",
pages = "523 - 532",
year = "1996",
issn = "0952-1976",
doi = "DOI: 10.1016/0952-1976(96)00049-8",
url = "http://www.sciencedirect.com/science/article/B6V2M-3WC4CC5-5/2/7a00b8f29ed3
author = "Il-kwon Jeong and Ju-jang Lee",
keywords = "Genetic algorithm",
keywords = "simulated annealing",
keywords = "system identification"
}
```

Genetic Algorithms (GAs) are search methods based on natural selection and genetics, while neural networks and fuzzy theory originate from human information processing and inference procedures. GA;s are currently being used in various problems, including control problems. In the control area, the GA has been used in identification, adaption and neural-network controllers.

GA is different from conventional optimization methods in several ways. The GA is a parallel and global search technique that search multiple points,

so it is more likely to obtain a global solution. It makes no assumption about the search space, so it is simple and can be applied to various problem. However, GAs are inherently slow, and are not good a fine-tuning solutions.

A Ga is a search method based on natural selection and genetics. The central theme of the research on GAs has been robustness, and balance between efficiency and efficacy necassary for survival in many different environments. GAs are computationally simple, yet powerful, and are not limited by assumptions about the search space.

The important thing to recognize ist hat a randomized search does not necessarily imply a directionless search. If more-human-like optimization tools are needed, the most important goal of optimization should be improvement. Although a GA cannot gaurantee that the solution will converge to the optimum, it tires to find the optimum, that is, it words towards an improvement. GAs are different from normal search procedures in four ways:

- GAs work with a coding of the parameter set, not the parameters themselves.
- GAs search from a population of points, not a single point.
- GAs use objective function information, not derivatives or otehr auxilary knowledge.
- GAs use probabilisistic transition rules, not deterministic rules.

The GA may be thought of as an evolutionary process, where a population of solutions evolves over a sequence of generations. During each generation, the fitness(goodness) of each solution is calculated, and solutions are selected for reproduction on the basic of their fitness. The probability of survival of a solution is proportional to its fitness value. This process is based on the principal of "survival of the fittest". The reproduced solutions then undergo recombination, which consists of crossover and mutation. A genetic representation may differ from the real form of the parameters of hte solutions. Fixed-length and binary encoded strings have been widely used for representing solutions, since they provide the maximum number of schemate, and are simple to implement.

A simple GA is really easy to use, yet powerful. It uses three basic genetic operators: reproduciton, crossover and mutation. Reproduction is a process in which individual strings (solutions) are copied according to their fitness values (objective function values). Crossover requires a mating of two rndomly selected strings. THe information on the string is partly

interchanged according to a randomly chosen crossover site. Crossover is applied to take valueable information from the parents, and it is applied with a certain probability. Mutation is the occasional random alteration of the value of a string position. Mutation insures against bit loss, and can be a source of new bits. Since mutation is a random walk thoruhg string space, it must be used sparingly.

There are three difference betwene GA and random searches. First, there exists a direction of the search, due to the selection probability. Second, the better strings generate more offspring. Finally, it is likely to be improved in average fitness after some generations.

4.1 Limitations of GA

The GA is a very useful algorithm because of its versatility. However, it has three major limitations. First, the perfromance is degraded as the problem size grows. As the problem size grows, the GA requires a larger population to obtain a satisfactory solution. This situation occurs, for example, when GA is used for the optimization of the weights or the structure of a large neural network. Secondly, premature convergence occurs when the GA cannot find the optimal solution due to loss of some important characters (genes) in the strings. The reason is that the GA depends heavily on crossover, and the mutation probability is generally too small to move the serach into another space. To overcome this problem, a large population can be used so that it includes many characters. However, this is not desirable because the computational burden is increased and the speed of convergence is slow. There has been much work done to prevent premature convergence for small populations: using the rank of the fitness values so that the selectivity is not proportional to the fitness value, scaling the fitness value according to the gene loss, changing the genetic operatorm cosntraining mating (incest prevention), lowering the fitness values of similar strings, adjusting the mutation probability or inserting new genes using parallel GA when the population is large, mergin the GA with another method, etc. Another limitation of the GA is that is lacks a hill-climbing capability. The reasion is also that the probability of mutation is much smaller that the probability of crossover.

4.2 Mutation operator

No change is made to the crossover operator of the GA. The proposed algorithm, named ASAGA, includes the merits of SA by changing the mutation

operator. This is basically different from only changing the operator probability. The new mutation operates like simulated annealing as follows. It generates a random stringm which it accepts if the string is better than the original string. Otherwise, the string is accepted according to the probability given in eqaution(1). For the cooling schedule which is related to the speed of ASAGA, equation (6) is used again. SA searches for the minimal energy state while the GA search for the string with the maximum fitness value.

5 A distributed hierarchical genetic algorithm for efficient optimization and pattern matching

```
@article{1221433,
 author = {Garai, Gautam and Chaudhuri, B. B.},
 title = {A distributed hierarchical genetic algorithm for efficient optimization
 journal = {Pattern Recogn.},
 volume = {40},
 number = {1},
 year = {2007},
 issn = {0031-3203},
 pages = {212--228},
 doi = {http://dx.doi.org/10.1016/j.patcog.2006.04.023},
 publisher = {Elsevier Science Inc.},
 address = {New York, NY, USA},
 }
```

Genetic Algorithm (GA) is a class of stochastic search methods inspired by the Darwinian's concept of sruvival of the fittest individual in natural selection and hence catergorized as a class of evolutionary algorithms. The technique was first formalized by Holland for use in adaptive systems. It has attracted a great deal of attention from researchers in numerous field as a way of effective and efficient search for optimization in complex, multi-dimensional space. GA represents a parallel adaptive search process which is executed with modification of genetic parameters in a wide variety of problems.

Such evolutionary computation technique try to iteratively reach the optimial solution of a problem closest to the actual or global solution. The algorithm stars with a set of individuals (called *population* and advances towards the solution with three basic operations namely, reproduction, crossover and

mutation. However, there is always a possibility that hte solution vector gets stuck in a local optimum. Several modified algorithsms have been proposed and implemented to jump out of this optima. One possibility is to maintain the diversity in the population and/or to distribute the individuals all over the search space as unifromly as possible. However, this usually leads to increase in the computational complexity of the system.

In the simple sequential GAs there is a possibility that after a few iteratinos the GA search may be confined toa local optimum. This may happen due to the dearth of diversity in the population for which the search cannot explore the entire space uniformly. Thus, the best solution obtained so far is not updated for several consecutive generations and the search is eventually terminated without reaching the global optimum. This nature is exhibited either for the deviation of the GA from the location of the global optimum solution or due to the inefficiency of hill climbing. The problem can, however, be either eliminated or reduced if the search is distributed uniformly as far as possible over the entire space.

6 A family of genetic algorithm packages on a workstation for solving combinatorial optimization problems

```
@article{182071,
 author = {Wainwright, Roger L.},
 title = {A family of genetic algorithm packages on a workstation for solving comb
 journal = {SIGICE Bull.},
 volume = {19},
 number = {3},
 year = {1994},
 issn = {0893-2875},
 pages = {30--36},
 doi = {http://doi.acm.org/10.1145/182063.182071},
 publisher = {ACM},
 address = {New York, NY, USA},
 }
```

In theory, the algorithms in the calss of NP-complete problems are considered computationaly equivalent. NP-complete problems have no known deterministic polynomial time algorithms; that is, there is no known solu-

tion except to try all combinations. These problems are teime referred to as combinatorial optimization problems. it is curretnly impossible to optimally sovle any of these problems, except for trivial cases. In general, finding a solution requires an organized search through the problem space. An unguided search is extremely inefficient. Consequently researchers have focused on approximation techniques which provide efficient, near optimal solutions.

6.0.1 Genetic Algorithm

There are several wasy that genetic algorithms differ from traditional algorithms. Genetic algorithsm are based on the principles of natural genetics and survial of the fittest. Genetic algorithms search through a solution space by emulating biological selection and reproduction. The GA works from a population of strings instead of a single point. The genetic algorithm works with a coding of parameter rather than the actual parameter. The genetic algorithm uses probabilistic transition rules, not deterministic rules, and applications of the genetic operators causes information fomr the previous generation to be carried over to the next. In addition, genetic algorithms produce "close" to optimal results in a "reasonable" amount of time. Futhermore, genetic algorithms are fairly simple to develop and they are suitable for parallel processing.

The genetic algorihtm is a robust search and optimization technique using probabilistic rules to evolve a population from one generation to the next. The transition rules going from one generation to the next are called genetic recombination operators. These include *Reproduction* (of the more fit chromosomes), Crossover, where portions of the two chromosomes are exchanged in some manner, and Mutation. Crossover combines the "fittest" chromosomes and passes superior genes to the next generation thus providing new points in the solution space. Mutation is performed infrequently. A new individual (point in the solution space) is created by altering some of the bits of an individual. Mutation ensures the entire state space will eventually be search (given enough time), and can lead the population out of a local minima. Genetic algorithms retain information from one generation to the next. This information is ued to prune the search space and generate plausible solution within the specified constraints.

The genetic algorihtm creates an initial population of feasible solutions, and then recombines them in such a way to guide the search to only the most promising areas of the state space. In a *generational* GA the offspring are saved in a serparate pool until the pool size is reached. Then the children's

pool replaces the parent's pool for the next generation. In a *steady-state* GA the offspring and parents occupy the same pool. Each time an offspring is generated iit is placed , and the weakest chromosome is "dropped off" the pool. These two cases represent two extremes in pool management for genetic algorithms.