

```

PROCEDURE Initialisation(Z, D, g, p_size);
/* generates a p_size population of strings */
BEGIN
    FOR j = 0 to (p_size-1) DO
        population[j] ← Generate(Z, D, g, j);
    END
    return(population);
END /* Initialisation */

PROCEDURE Generate(Z, D, g, j);
/* create string length = (fitness value+|Z| values+|Z| binary template values) */
BEGIN
    FOR i = 1 to |Z| DO
        population[j,i] ← Rand(1, |Di|);
        population[j,(i+|Z|)] ← Rand(0, 1);
    END
    population[j,0] ← Fitness(g, population, j);
    return(population);
END /* Generate */

PROCEDURE Reproduction(Z, g, population, p_size, elite);
/* biased selection of parents from the population for mating in Crossover */
BEGIN
    matepool ← Elitism(Z, g, population, p_size, elite);
    FOR j = elite to (p_size-1) DO
        matepool[j] ← Select(Z, g, matepool, population, p_size, j);
    END
    return(matepool);
END /* Reproduction */

PROCEDURE Elitism(Z, g, population, p_size, elite);
/* selection of elite best population members for matepool */
BEGIN
    population ← Sort(population, ascending);
    FOR j = 0 to (elite-1) DO
        FOR i = 0 to (2 * |Z|) DO
            matepool[j,i] ← population[j,i];
        END
    END
    return(matepool);
END /* Elitism */

PROCEDURE Select(Z, g, matepool, population, p_size, j);
/* biased selection of population member using roulette wheel selection */
BEGIN
    FOR j = 0 to (p_size-1) DO
        p_fitness[j] ← p_fitness[j] + Fitness(g, population, j);
    END
    target ← Random() * p_fitness;
    member ← 0;
    total ← 0;
    REPEAT
        total ← total + Fitness(g, population, member);
        IF total ≤ target THEN member ← member + 1;
        IF member = p_size THEN member ← 0;
    UNTIL (total > target);
    FOR i = 0 to (2*|Z|) DO
        matepool[j,i] ← population[member,i];
    END
END

```

```

        END
        return(matepool);
    END /* Select */

PROCEDURE Crossover(Z, D, C, g, matepool, p_size, n_ospring, hc_time);
/* each offspring is created by exchanging parents' genetic material using their binary templates */
BEGIN
    FOR j = 1 to n_ospring DO
        matepool ← UAX(Z, g, matepool, p_size, j);
        IF C ≠ {} THENmatepool ← Repair(Z, D, C, g, matepool, (p_size- j));
        IF hc_time > 0 THEN matepool ← HC(Z, g, matepool, hc_time, (p_size-j));
        END
    END
    return(matepool);
END /* Crossover */

PROCEDURE UAX(Z, g, matepool, p_size, j);
/* create offspring by exchanging two parents string values using their binary templates */
BEGIN
    p1 ← Rand(0, (p_size-1));
    p2 ← Rand(0, (p_size-1));
    p ← p1;
    FOR i = 1 to |Z| DO
        IF (matepool[p1,(i+|Z|)] = matepool[p2,(i+|Z|)] AND p = p1) THEN p ← p2;
        IF (matepool[p1,(i+|Z|)] = matepool[p2,(i+|Z|)] AND p = p2) THEN p ← p1;
        offspring[i] ← matepool[p,i];
        offspring[i+|Z|] ← matepool[p,(i+|Z|)];
    END
    offspring[0] ← Fitness(g, offspring, 0);
    FOR i = 0 to (2 * |Z|) DO
        matepool[(p_size-j),i] ← offspring[i];
    END
    return(matepool);
END /* UAX */

PROCEDURE Repair(Z, D, C, g, matepool, j);
/* make sure the correct number of each domain value in C are represented in offspring j */
BEGIN
    FOR i = 1 to |Z| DO
        values[matepool[j,i]] ← values[matepool[j,i]] + 1;
    END
    number_of_low_values = 0;
    FOR k = 1 to |D| DO
        IF values[k] < number of required cars of this type, as specified in C THEN
            number_of_low_values ← number_of_low_values + 1;
            under[number_of_low_values] ← k;
        END
    END
    FOR k = 1 to number_of_low_values DO
        FOR m = 1 to (number of required cars of this type -values[under[k]]) DO
            b_pos ← 0;
            FOR i = 1 to |Z| DO
                IF values[matepool[j,i]] > number of required cars of this type THEN
                    val_over ← matepool[j,i];
                    matepool[j,i] ← under[k];
                    matepool[j,0] ← Fitness(g, matepool, j);
                    IF matepool[j,0] < b_fitness OR b_pos = 0 THEN
                        b_fitness ← matepool[j,0];
                        b_pos ← i;
                    END
                END
            END
        END
    END
END

```

```

        END
        matepool[j,i] ← val_over;
    END
    END
    values[matepool[j,b_pos]] ← values[matepool[j,b_pos]] - 1;
    values[under[k]] ← values[under[k]] + 1;
    matepool[j,b_pos] ← under[k];
    matepool[j,0] ← b_fitness;
    END
    END
END /* Repair */

PROCEDURE HC(Z, g, matepool, hc_time, j);
/* for hc_time CPU seconds exchange expensive val_a's with val_b's which reduce the offspring
fitness */
BEGIN
    WHILE hc_time > Elapsed()
        pos_a ← Fitness(g, matepool, j);
        b_pos ← 0;
        FOR i = 1 to |Z| DO
            IF matepool[j,i] ≠ matepool[j,pos_a] THEN
                val_b ← matepool[j,i];
                matepool[j,i] ← matepool[j,pos_a];
                matepool[j,pos_a] ← val_b;
                matepool[j,0] ← Fitness(g, matepool, j);
                IF matepool[j,0] << b_fitness OR b_pos = 0 THEN
                    b_fitness ← matepool[j,0];
                    b_pos ← i;
                END
                matepool[j,pos_a] ← matepool[j,i];
                matepool[j,i] ← val_b;
            END
        END
        val_b ← matepool[j,b_pos];
        matepool[j,b_pos] ← matepool[j,pos_a];
        matepool[j,pos_a] ← val_b;
        matepool[j,0] ← b_fitness;
    END
END /* HC */

PROCEDURE Elapsed();
/* return the number of CPU seconds elapsed */

PROCEDURE Fitness(g, population, j);
/* return the g-value (g is the objective function) of the j-th member of the population */

PROCEDURE Fitness(g, population, j);
/* return a high cost j string value */

PROCEDURE Rand(lower, upper);
/* return a random number between lower and upper */

PROCEDURE Random();
/* return a random number between 0.00 and 1.00 */

PROCEDURE Sort(population, ascending);
/* sort population in ascending fitness order */

```

Pseudo Codes for HR and Tabu:

(Z, D, C) is a CSP; in our tests, IterationLimit was set to 100,000 and TabuLengthLimit was set to 1.

```
PROCEDURE TABU_CSP_1(Z, D, C, IterationLimit, TabuLengthLimit)
BEGIN
  FOR each variable xi in Z (random order) DO
    x  $\leftarrow$  any value from Dx;
    Tabu[i]  $\leftarrow$  empty list;
  END;
  i  $\leftarrow$  1;
  REPEAT
    S  $\leftarrow$  set of variables which label violates some constraints;
    pick a random variable y from S;
    v  $\leftarrow$  value currently assigned to y;
    V  $\leftarrow$  set of values in Dy which are not in Tabu[y];
    y  $\leftarrow$  the value which in V which involves in the least number of conflicts, break ties
      randomly;
    Make v the last element of Tabu[y];
    IF (the number of elements in Tabu[y]  $\geq$  TabuLengthLimit)
      THEN Remove the first element from Tabu[y];
    UNTIL i  $\geq$  IterationLimit;
  END /* of TABU_CSP_1 */
```

When TabuLengthLimit = 0, TABU_CSP_1 becomes a HR algorithm;
when TabuLengthLimit = 1, TABU_CSP_1 is a one-state Tabu algorithm.

References

- Bagchi, S., Uckun, S., Miyabe, Y. & Kawamura, K., *Exploring Problem-Specific Recombination Operators for Job Shop Scheduling*, Proc., Fourth International Conference on Genetic Algorithms, 1991, 10-17
- Chew, T.L., David, J.M., Nguyen, A., & Tourbier, Y., *Solving constraint satisfaction problems with simulated annealing: the car sequencing problem revisited*, Proc., International Workshop on Expert Systems & Their Applications, Avignon, France, 1992, 405-416
- Cleveland, G.A. & Smith, S.F., *Using Genetic Algorithms to Schedule Flow Shop Releases*, Proc., Third International Conference on Genetic Algorithms, 1989, 160-169
- Cras, J-Y., *A review of industrial constraint solving tools*, AI Perspective Series, AI Intelligence, Oxford, UK, 1993

Davenport, A., Tsang, E.P.K., Wang, C.J. & Zhu, K., *GENET: a connectionist architecture for solving constraintsatisfaction problems by iterative improvement*, Proc., 12th National Conference for Artificial Intelligence (AAAI), 1994, 325-330

Davenport, A. & Tsang, E.P.K., *Solving constraint satisfaction sequencing problems by iterative repair*, Proceeding, 14th UK Planning and Scheduling Special Interest Group Workshop, November, 1995 (to appear)

De Jong, K.A., *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*, Doctoral dissertation, University of Michigan, Department of Computer and Communication Sciences. Dissertation Abstracts International 36 (10), 5140B, 1975 (University Microfilms No. 76-9381)

De Jong, K. & Spears, W., *On the state of evolutionary computation*, Proc., Fifth International Conference on Genetic Algorithms, 1993, 618-623

Dincbas, M., Simonis, H., & Van Hentenryck, P., *Solving the Car Sequencing Problem in Constraint Logic Programming*, Proc., European Conference on Artificial Intelligence, 1988, 290-295

Eiben, A. E., Raue, P.E., & Ruttkay, Zs., *GA-easy and GA-hard Constraint Satisfaction Problems*, Workshop on Constraint Processing, 11th European Conference on Artificial Intelligence, 1994, 87-96

Eiben, A. E., Raue, P.E., & Ruttkay, Zs., *Solving constraint satisfaction problems using genetic algorithms*, Proc., IEEE World Conference on Computational Intelligence, 1st IEEE Conference on Evolutionary Computation, 1994, 543-547

Filipic, B., *Enhancing genetic search to scheduling a production unit*, Proc., 10th European Conference on Artificial Intelligence, 1995, 603-607

Fang H-L., Ross P. & Corne D., *A Promising Genetic Algorithm Approach to Job-Shop Scheduling, Re-scheduling and Open-Shop Scheduling Problems*, Proc., Fifth International Conference on Genetic Algorithms, 1993, 375-382

- Fleurent, C., & Ferland, J.A., *Object-oriented implementation of heuristic search methods for graph coloring, maximum clique, and satisfiability*, Trick, M.A. & Johnson, D.S. (eds.), DIMACS Series in Discrete Mathematics, 1995 (to appear)
- Fox, M., & Sadeh, N., *Why is scheduling difficult, a CSP perspective*, Invited talk, European Conference on AI, 1990, 754-767
- Freuder, E.C., & Mackworth, A. (eds.), *Constraint-based reasoning*, MIT Press, 1994
- Freuder, E.C., & Wallace, R.J., *Partial constraint satisfaction*. Artificial Intelligence, vol 58, Nos 1-3 (Special Volume on Constraint Based Reasoning), 1992, 21-70
- Freuder, E.C., Dechter, R., Ginsberg, M., Selman, B. & Tsang, E., *Systematic versus stochastic constraint satisfaction*, Proc., 14th International Joint Conference on AI, August, 1995, 2027-2032
- Gent, I.P., & Walsh, T., *An empirical analysis of search in GSAT*, Journal of Artificial Intelligence Research, 1993, 47-59
- Glover, F., *Tabu search Part I*. Operations Research Society of America (ORSA), Journal on Computing vol 1, 1989, 109-206
- Glover, F., *Tabu search Part II*. Operations Research Society of America (ORSA), Journal on Computing vol 2, 1990, 4-32
- Goldberg, D.E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Reading, MA, Addison-Wesley publishing Company Inc., 1989
- Goldberg, D.E., *Real-coded Genetic algorithms, Virtual Alphabets, and Blocking*, IlliGAL Report No. 90001, Department of General Engineering, University of Illinois, September, 1990
- Holland, J.H., *Adaptation in Natural and Artificial Systems*, Ann Arbor: The University of Michigan Press, 1975
- Kadaba N., Nygard K.E. & Juell P.L., *Integration of Adaptive Machine Learning and Knowledge-Based Systems for Routing And Scheduling Applications*, Proc., International Expert Systems with Applications Journal vol 2, 1991, 15-27

- Michalewicz Z. & Janikow C.Z., *Handling Constraints in Genetic algorithms*, Proc., Fourth International Conference on Genetic Algorithms, 1991, 151-157
- Michalewicz Z., Vignaux G.A. & Groves L.J., *Genetic Algorithms for Optimization Problems*, Proc., Eleventh New Zealand Computer Conference, Wellington, New Zealand, August 16-18, 1989, 211-223
- Minton, S., Johnston M.D., Philips, A.B., & Laird, P. (1990). *Solving Large-Scale Constraint Satisfaction and Scheduling Problems Using a Heuristic Repair Method*, Proc., Eighth National Conference on Artificial Intelligence, Menlo Park, California, USA: American Association for Artificial Intelligence. 17-24
- Naden, P., *Constraint satisfaction using tabu search*. MSc Dissertation, Department of Computer Science, University of Essex, 1994
- Parrello, B.D., *CAR WARS: AI Expert*, 1988, 60-64
- Parrello, B.D., Kabat, W.C., & Wos, L., *Job-shop scheduling using automated reasoning: a case study of the car sequencing problem*, Journal of Automatic Reasoning, vol 2, no 1, 1986, 1-42
- Richardson J.T., Palmer M.R., Liepins G.E. & Hilliard M.R., *Some Guidelines for Genetic Algorithms with Penalty Functions*, Proc., Third International Conference on Genetic Algorithms, 1989, 191-197
- Schaffer, D.J., & Morishima, A., *An Adaptive Crossover Distribution Mechanism for Genetic algorithms*, Proc., Second International Conference on Genetic Algorithms, 1987, 36-40
- Selman, B., Levesque, H. & Mitchell, D., *A new method for solving hard satisfiability problems*, Proc., National Conference on Artificial Intelligence (AAAI), 1992, 440-446
- Selman, B. & Kautz, H., *Domain-independent extensions to GSAT: solving large structured satisfiability problems*, Proc., 13th International Joint Conference on AI, 1993, 290-295
- Selman, B., Kautz, H.A. & Cohen, B., *Noise strategies for improving local search*, Proc., 12th National Conference for Artificial Intelligence (AAAI), 1994, 337-343

- Siedlecki W. & Sklansky J., *Constrained Genetic Optimization via Dynamic Reward-Penalty Balancing and Its use in Pattern Recognition*, in Schaffer, J.D. (ed.), Proc., Third International Conference on Genetic Algorithms, 1989, 141-150
- Syswerda, G., *Uniform Crossover in Genetic algorithms*, Proc., Third International Conference on Genetic Algorithms, 1989, 2-9
- Tsang, E.P.K., & Warwick, T., *Applying Genetic algorithms to Constraint Satisfaction Optimisation Problems*, Proc., European Conference on Artificial Intelligence, Stockholm, Sweden, 1990, 649-654
- Tsang, E.P.K., *Foundations of Constraint Satisfaction*, Academic Press, London and San Diego, 1993
- Wallace, R.J., & Freuder, E.C., *Conjunctive width heuristics for maximal constraint satisfaction*. Proc., National Conference on Artificial Intelligence (AAAI), 1993, 762-768
- Warwick, T., & Tsang, E.P.K, *Using a Genetic Algorithm to Tackle the Processors Configuration Problem*, Proc., ACM Symposium on Applied Computing, 1993, 217-221
- Warwick, T., *A GA Approach to constraint satisfaction problems*, PhD Thesis, Department of Computer Science, University of Essex, Colchester, UK, 1995
- Whitley, D., Starkweather, T. and Shaner, D., *The Traveling Salesman and Sequence Scheduling: Quality Solutions Using Genetic Edge Recombination*, in Handbook of Genetic Algorithms (ed. L. Davis), New York, USA: Van Nostrand Reinhold, 1991, 350-372
- Zhu, K., 1993, Unpublished results in the GENET project, EPSERC funded (UK), reference GR/H75275, (Dr. Zhu was a visiting scholar at the University of Essex in April - November 1993)