Miscellaneous

Daniel Lopes Cinalli

Universidade Federal Fluminense, Niterói, Brazil

Table of Contents

1 Mathematical Model

1.1 Bi-objective

The problem is formally represented as:

$$\min \sum_{i=1}^{N} \sum_{j=1}^{M} \sigma_{ij} d_{ij} + \sum_{j=1}^{M} c_j \mu \tag{1}$$

$$\max \sum_{i=1}^{N} \sum_{j=1}^{M} \sigma_{ij} v_j \text{ (or) } \max \sum_{i=1}^{N} \sum_{j=1}^{M} \sigma_{ij} h(v_j, a_i) \tag{2}$$

Let μ be the cost of one processing unit, v the productive capacity of one processing unit linked to one resource area, a the available resource for the area, h a function to outcome the minimum of two values, M a set of available positions to processing units, N a set of available positions to resource areas and D a distance matrix $(d_{ef})_{n_x m}$, where $n \in N$ and $m \in M$. The decision variables are the processing unit c_j ($j \in M$) that assumes 1 if it is placed at position j or 0 otherwise and σ_{ij} that assumes 1 if there is a link between the resource area g at position $i \in N$ and the processing unit at position $j \in M$.

$$c_j = \begin{cases} 1 & \text{if the processing unit is placed} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

$$\sigma_{ij} = \begin{cases} 1 & \text{if there is a link between } g_i \text{ and } c_j \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

1.2 Tri-objective

The problem is formally represented bellow:

$$\min \sum_{i=1}^{N} \sum_{j=1}^{M} \sigma_{ij} d_{ij} \tag{5}$$

$$\min \sum_{j=1}^{M} c_j \mu \tag{6}$$

$$\max \sum_{i=1}^{N} \sum_{j=1}^{M} \sigma_{ij} v_j \text{ (or) } \max \sum_{i=1}^{N} \sum_{j=1}^{M} \sigma_{ij} h(v_j, a_i) \tag{7}$$

1.3 Constraints

Let P the available positions to one specific area, q the maximum number of resource areas allowed, k the maximum number of processing unit allowed and r the minimum distance between the processing unit and the resource area. The constraints are defined as:

$$1 \leq k \leq q \tag{8}$$

$$1 \leq \sum_{j=1}^{M} c_j \leq k \tag{9}$$

$$\sum_{i=1}^{P} \sigma_{ij} = 1, j = 1, \ldots, M \tag{10}$$

$$\sum_{i=1}^{N} \sum_{j=1}^{M} \sigma_{ij} \geq r \tag{11}$$

Considering the link σ_{ij} between one resource area g_i and the processing unit c_j, a geometric interpretation of σ_{ij} transforms the link into a 2-dimensional space line segment $\overline{g_i c_j}$ defined by two distinct points (x_1, y_1) for g_i and (x_2, y_2) for c_j. Another constraint to the problem states that the links (lines segments) in the solution cannot intercept each others. Therefore, given two links and their respective points: (x_1, y_1) and (x_2, y_2) for $\overline{g_i c_i}^A$; (x_3, y_3) and (x_4, y_4) for $\overline{g_i c_i}^B$; the intersection P of two lines segments is verified if each of the two pairs of cross products below have different signs (or one cross product in the pair is 0). The Sweeping algorithm [1] deals with a set of n line segments in the plane and avoids testing pairs of segments that are far apart.

$$(p_1 - p_4) \times (p_3 - p_4) \;\; \& \;\; (p_2 - p_4) \times (p_3 - p_4) \tag{12}$$

$$(p_3 - p_2) \times (p_1 - p_2) \;\; \& \;\; (p_4 - p_2) \times (p_1 - p_2) \tag{13}$$

1.4 Data Structure

In genetic algorithms, the individual structure is referred to as genotype, gene or chromosome and the values that a chromosome may take on are named as alleles. The computational representation for the problem in section 1.1 considers the processing unit as a tuple $c_i =< x, y, t >$; where $c_i \in C = \{c_1, ..., c_k\}$, t is the type of the unit, x and y are the Cartesian coordinates of the position. The resource area is represented by the tuple $a_i =< x, y, l >$; where $a_i \in A = \{a_1, ..., a_q\}$, l is a index that links the resource area a_i to the unit c_i. Thus, the chromosome encoding is the aggregation of these tuples regulated by q resource areas and k processing units.

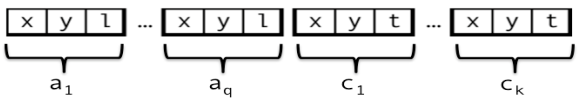

Fig. 1. Chromosome encoding for the problem.

2 Foundations

Evolutionary Alg.

Evolutionary algorithms (EA) are inspired in Darwinian principles of natural evolving systems to automated problem solving [2]. The methods are population-based and cover a wide range of problems that includes constraint satisfaction and combinatorial optimization. Its typical components are a space of individuals I to be considered as candidate solutions, a problem-specific fitness function of individuals $F : I \to \mathbb{R}$ and some mechanisms for search strategy. The underlying idea behind EAs can be viewed as a crossover-mutation-fitness evaluation-selection loop until a termination criterion is reached, such as: a candidate with acceptable quality or a previous computational constraint. High-level framework uses μ and λ to represent parent and offspring population sizes. Therefore, a population P of individuals $a \in I$ at generation t, $P(t) = \big(a_1(t), \ldots, a_\mu(t)\big) \in I^\mu$, applies selection, mutation, and crossover operators respectively described as: $s : I^\lambda \to I^\mu$, $m : I^k \to I^\lambda$ and $r : I^\mu \to I^k$.

Hypervolume

The quality of MOEAs' trade-off fronts can be quantitatively evaluated by many different techniques. The hypervolume or S-metric [3, 4] is a quantitative measure of the space covered by the non-dominated points. It represents the union of hypercubes a_i defined by a non-dominated point m_i and a reference point x_{ref} as defined in (14).

$$\begin{aligned} S(M) &= \Lambda\left(\{\textstyle\bigcup_i a_i | m_i \in M\}\right) \\ &= \Lambda\left(\textstyle\bigcup_{m \in M}\{x | m \prec x \prec x_{ref}\}\right). \end{aligned} \tag{14}$$

A second measure widely used is the coverage of two sets [4];

$$C(X', X'') = \frac{|\{a' \in X''; \exists a' \in X' : a' \prec a''\}|}{|X''|}, \tag{15}$$

where $X', X'' \subseteq X$ are two sets of decision vectors and function C maps the percentage of domination from one set to another in the interval $[0, 1]$. Although convex regions may be preferred by the hypervolume, it can be used independently to evaluate the performance of different multi-objective algorithms. The coverage of two sets technique has no restriction related to the shape of Pareto front, but it does not express how much better one set is over the other.

Mixture Model

A mixture model is a probabilistic model to reveal distributions of observations in the overall population [5, 6]. Given a data set $D = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ where $\mathbf{x}_i$ is a d-dimensional vector measurement with the points created from density $p(\mathbf{x})$, a finite mixture model is defined as:

$$p(\mathbf{x}|\Theta) = \sum_{k=1}^{K} \alpha_k p_k(\mathbf{x}|z_k, \theta_k) \tag{16}$$

Let $K \geq 1$ be the number of components, $p_k(\mathbf{x}|z_k, \theta_k)$ be the mixture components where each k is a density or distribution over $p(\mathbf{x})$ and parameters θ_k, $\mathbf{z} = \{z_1, \ldots, z_k\}$ be a K-ary random variable defining the identity of the mixture component that produced $\mathbf{x}$ and $\alpha_k = p_k(z_k)$ are the mixture weights representing the probability that $\mathbf{x}$ was generated by component k. Hence, the parameters for a mixture model is $\Theta = \{\alpha_1, \ldots, \alpha_K, \theta_1, \ldots, \theta_K\}$, $1 \leq k \leq K$.

In a Gaussian mixture model, each of the K components is a Gaussian density with parameters $\theta = \{\mu_k, \Sigma_k\}$, $\mathbf{x} \in \mathbb{R}^d$ and function as:

$$p_k(\mathbf{x}|\theta_k) = \frac{1}{(2\pi)^{d/2}|\Sigma_k|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\mu_k)^t \Sigma_k^{-1}(\mathbf{x}-\mu_k)} \tag{17}$$

Expec. Maximization

The expectation maximization (EM) algorithm [7] for Gaussian mixture is a particular way of implementing the maximum likelihood estimation in probabilistic models with incomplete or missing data values. EM learns the parameters θ_k guessing a distribution for the unobserved data and finds the cluster to which a singular chromosome most likely belongs. It starts with an initial estimation of Θ and iterates between E-step and M-step of the algorithm to update Θ until convergence.

E-step estimates the posterior distribution of the latent variables taking into account the current parameters and the observed data. The membership weight w_{ik} computes the probability of all data points $\mathbf{x}_i$ to the mixture components k. In the M-Step, the algorithm uses the calculated membership weights to find new model parameters values.

$$w_{ik} = p(z_{ik} = 1|\mathbf{x}, \Theta) = \frac{p_k(\mathbf{x}|z_k, \theta_k)\alpha_k}{\sum_{m=1}^{K} p_m(\mathbf{x}|z_m, \theta_m)\alpha_m}, 1 \leq k \leq K, 1 \leq i \leq N \tag{18}$$

After E and M steps the convergence is computed using the value of the log-likelihood $\log l(\Theta)$. The algorithm stops when there are no significant changes in the convergence from one iteration to the next.

$$\log l(\Theta) = \sum_{i=1}^{N} \log p(\mathbf{x}|\theta) = \sum_{i=1}^{N} \left(\log \sum_{k=1}^{K} \alpha_k p_k(\mathbf{x}|z_k, \theta_k) \right) \tag{19}$$

3 Results of EMOA (without COIN integration)

Using the 20x20 and 50x50 scenarios, tables 1 and 2 show the best results after 40 independent executions per EA. Every MOEA's execution performed 400 generations for 200 individuals (population) in the 20x20 domain and 900 generations for 700 individuals in 50x50. Each EA's execution returns a Pareto front F based on the union of all previous non-dominant sets throughout the generations: $F_i = nd(Pi \cup F_{i-1})$; where P_i is the set of points for generation i, F_{i-1} is the Pareto front from the previous one and nd is a dominance function. Considering all Pareto fronts produced ($m = 40$, in this experiment) and h the hypervolume function, $\bar{H} = \frac{\sum_{i=1}^{m} h(F^i)}{m}$ is the average hypervolume used to analyse the best operators and algorithm for the problem.

The crossover rate is 0.30 and the mutation rate is 0.10. Due to the problem's constraints, after those operations some new individuals might have invalid combinations of units and gates positions. Four strategies were implemented to repair the offspring (see section A): *fix random* - which randomly repairs gates' position, units' position and links; *fix greedy* - which repairs units' position by the closest distance to the area; *no fix loop* - does not repair, but rejects the invalid individual and keeps selecting new individuals until population is complete; *no fix* - does not repair any individual, just assigns a bad fitness value to it.

Table 1. 20x20 average hypervolume analysis for *fix randon* strategy

	Crossover	Mutation	Selection	Hypervolume
$\bar{H}$	Two-Point			0.5922
	One-Point		NSGA-II	0.5795
	Uniform			0.5939
	Uniform	Flip Bit	NSGA-II	0.5990
		Uniform Int		**0.6021**
		Shuffle Indexes		0.6005

Table 2. 20x20 average hypervolume analysis for all strategies

	Fix	Selection	Hypervolume
$\bar{H}$	Randon	NSGA-II	**0.6021**
		SPEA-2	0.5874
	Greedy	NSGA-II	0.6025
		SPEA-2	0.5932
	No Fix Loop	NSGA-II	0.5983
		SPEA-2	0.5494
	No Fix	NSGA-II	0.5815
		SPEA-2	0.5372

Figure 2 reports the distribution of results comparing different multi-objective genetic algorithms. For this problem, NSGA-II returned better outcomes than SPEA2 and the values are more concentrated to the mean. Figure 3 demonstrates two solutions from Pareto-optimal set composed of six resource areas (gray rectangles) and processing units (coloured circles). All resource areas are connected to only one unit in an optimal arrangement regarding the two objectives: cost and production.

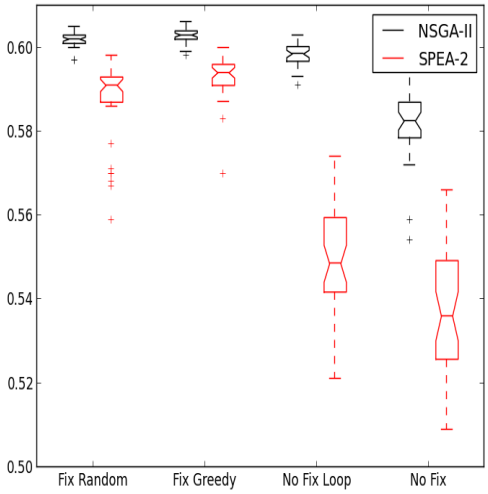

Fig. 2. 20x20 distribution of results after simulations

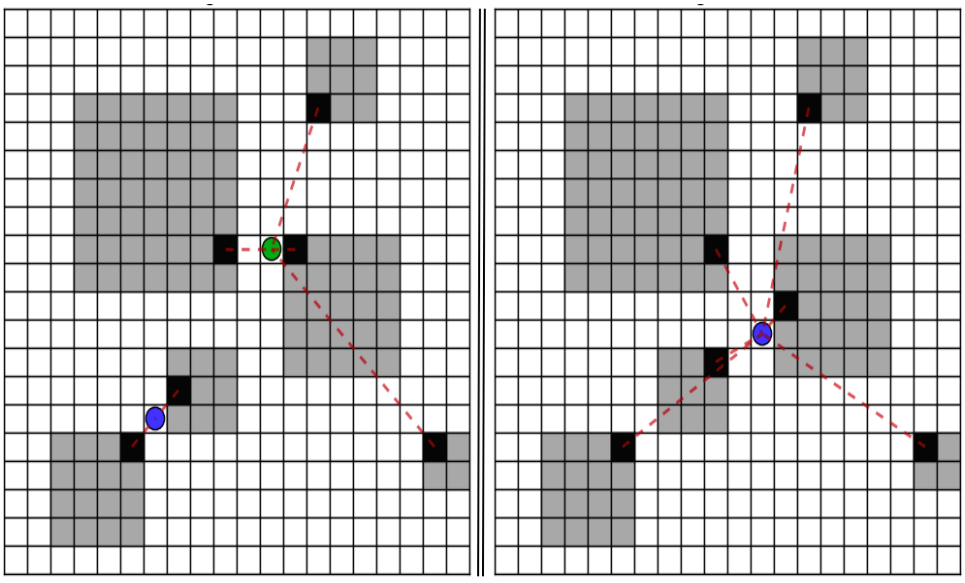

Fig. 3. Evolutionary algorithm solution for six resource areas.

3.1 Pareto-optimum front

The Pareto-optimal front for this real-world problem is calculated after 140 independent executions per EA. Considering all Pareto fronts P^i achieved, P^* is the set resulted after the dominance function nd over all P^i, such that: $P^* = nd(P^i), i = 1, \ldots, m$ (in this case, $m = 140$). The best non-dominated set P^b from all executions is compared to P^* in terms of convergence and diversity [8]. The experiment used the NSGA-II algorithm, uniform crossover and uniform integer mutation because this configuration achieved the best results in table 1.

Table 3. Comparation between 20x20 Pareto-optimum front and the best MOEA's solution

	Hypervolume	Optimum Convergence	Optimum Diversity
P^*	0.6080	—	—
P^b	0.6058	0.7835	1.0322

Table 4. Comparation between 50x50 Pareto-optimum front and the best MOEA's solution

	Hypervolume	Optimum Convergence	Optimum Diversity
P^*	0.7568	—	—
P^b	0.7565	0.0212	1.5378

4 Algorithm

ICIEA implements collective pairwise comparisons of scenarios into the optimization search for the resource placement problem explained in section 1.1. The approach iteratively suspends the evolution to ask users for their preferred individuals in the population. Based on the group's subjectivity and cognition, all the population fitness values are recalculated and reinstated for next generations of MOEA. The algorithm is presented as follows:

Algorithm 1 ICIEA algorithm

```
generation ← numgeneration                          ▷ number of generations
block ← subsetgeneration                            ▷ number of generations before pairwise
while generation do
    while block do
        offspring ← Tournament(pop)                 ▷ based on dominance
        offspring ← Crossover(offspring)
        offspring ← Mutation(offspring)
        pop ← NSGA-II(offspring)
    end while
    comparisons ← CollectivePairWise(F^i)           ▷ F^i is the actual Pareto front
    Θ ← ExpectationMaximization(comparisons)        ▷ Θ = {θ_pro, θ_anti}
    pop ← FitnessRecalculation(pop, Θ)
end while
```

Considering the opposed objectives of cost and production (section 1.1), the collective pairwise comparisons represent sample observations that highlight points in the search space throughout the evolution. In this case, however, there is an unobserved latent factor. The selected individuals may be most likely associated with a pro-cost or anti-cost (pro-production) cluster, but, as the comparisons are anonymous, the previous tendencies and the distribution of the unobserved data are completely unknown. Hence, the EM algorithm finds a mixture of one-dimensional Gaussian distributions for $K = 2$ components: $\theta_{pro}, \theta_{anti}$. The expression patterns for each cluster define the membership of the observations and use that probability to adjust a certain amount Ψ to the fitness values. Algorithm 2 presents the fitness recalculation.

Algorithm 2 ICIEA Fitness Recalculation

```
procedure FitnessRecalculation(pop, Θ)
    θ_pro ← Θ[0]                                    ▷ θ = {μ, σ²}
    θ_anti ← Θ[1]
    for all individual in pop do
        cost ← individual_fitness[cost]
        p_cost ← GaussianDensity(θ_pro, cost)       ▷ membership to θ_pro
        Ψ =| p_cost.σ²_pro.zscore(cost, θ_pro) |
        individual_fitness[cost] += Ψ

        anticost ← individual_fitness[prod]
        p_anticost ← GaussianDensity(θ_anti, anticost)   ▷ membership to θ_anti
        Ψ =| p_anti.σ²_anti.zscore(anticost, θ_anti) |
        individual_fitness[prod] += Ψ
    end for
    return pop
end procedure
```

Unlike K-means where data point must exclusively belong to one cluster center, Gaussian mixture model offers the possibility of many association for one unique point and this characteristic benefits the increase strategy of the fitness for distinct objectives.

A Appendix: Fix Algorithms

Four strategies were implemented to repair the offspring after genetic operations. Their algorithms are present as follows: *fix random* in algorithm 3 - which randomly repairs gates' position, units' position and links; *fix greedy* in algorithm 4 - which repairs units' position by the closest distance to the area; *no fix loop* in algorithm 5 - does not repair, but rejects the invalid individual and keeps selecting new individuals until population is complete; *no fix* in algorithm 6 - does not repair any individual, just assigns a bad fitness value to it.

Algorithm 3 ICIEA fix random

```
1:  procedure FIXRANDOM(invalids)
2:      for all individual in invalids do
3:          units ← GetUnits(individual)          ▷ processing units from chromosome
4:          areas ← GetAreas(individual)          ▷ resource areas from chromosome
5:
6:          for all position in units do                          ▷ fix processing units
7:              if wrong_position then
8:                  position ← new_position                              ▷ random
9:              end if
10:         end for
11:         for all position in areas do                          ▷ fix resource area
12:             if wrong_position then
13:                 position ← new_position
14:             end if
15:         end for
16:         for all area_link in areas do          ▷ fix links between area and unit
17:             if wrong_area_link then
18:                 area_link ← RandomUnitLink()
19:             end if
20:         end for
21:     end for
22: end procedure
```

Algorithm 4 ICIEA fix greedy

```
1:  procedure FIXGREEDY(invalids)
2:      for all individual in invalids do
3:          units ← GetUnits(individual)          ▷ processing units from chromosome
4:          areas ← GetAreas(individual)          ▷ resource areas from chromosome
5:
6:          for all position in units do          ▷ fix processing units
7:              if wrong_position then
8:                  position ← new_position       ▷ random
9:              end if
10:         end for
11:         for all position in areas do          ▷ fix resource area
12:             if wrong_position then
13:                 position ← new_position
14:             end if
15:         end for
16:         for all area_link in areas do         ▷ fix links between area and unit
17:             if wrong_area_link then
18:                 area_link ← ClosestUnitLink()
19:             end if
20:         end for
21:     end for
22: end procedure
```

Algorithm 5 ICIEA fix loop

```
1: still_invalids ← len(pop)
2: while still_invalids do
3:     offspring ← Tournament(pop)
4:     offspring ← Crossover(offspring)
5:     offspring ← Mutation(offspring)
6:     still_invalids ← Invalids(offspring)      ▷ number of invalids individuals
7: end while
8: pop ← NSGA-II(offspring)
```

Algorithm 6 ICIEA no fix

```
1: procedure NoFix(invalids)
2:     for all individual in invalids do
3:         individual_fitness[cost] ← ∞
4:         individual_fitness[prod] ← ∞
5:     end for
6: end procedure
```

B Appendix: ICIEA Gamification

Fig. 4. Gamification first screen

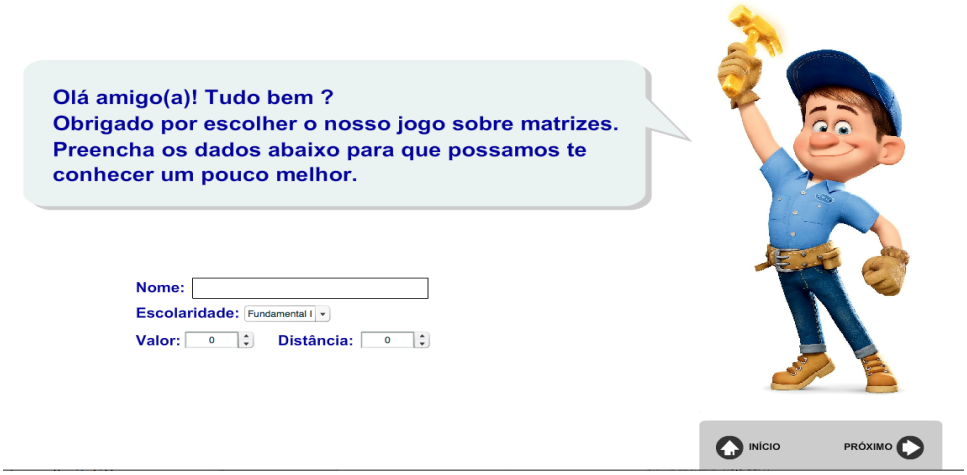

Fig. 5. User's profile

Fig. 6. User's profile

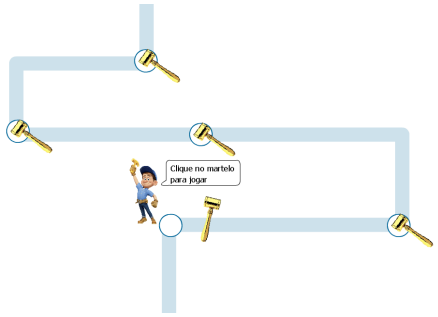

Fig. 7. Pairwise comparisons

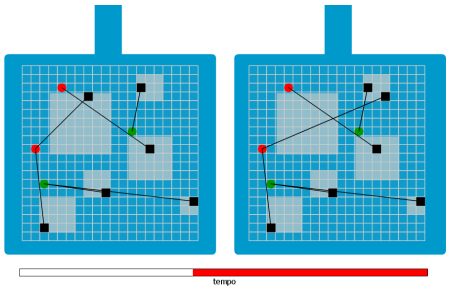

Fig. 8. Pairwise comparisons

C Appendix: ER model

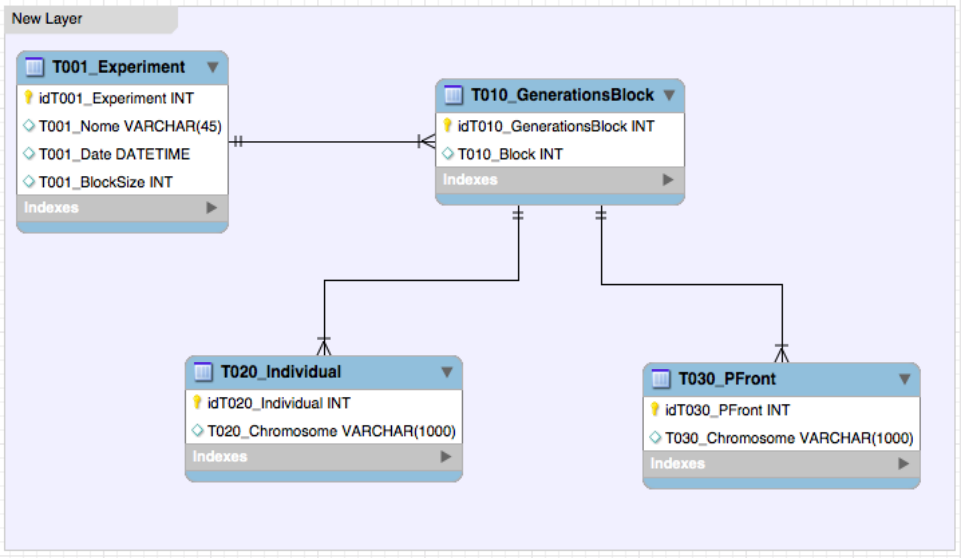

Fig. 9. ICIEA ER model

D Appendix: Multiattribute Utility

Formally represented as utility functions $U_1(x_1),...,U_m(x_m)$ to m different attributes x_1 to x_m. Each function takes 0 to the worst and 1 to the best level of an objective.

The utility function is:

$$U(x_1,...,x_m) = k_1U_1(x_1) + ... + k_mU_m(x_m) = \sum_{i=1}^{m} k_iU_i(x_i) \tag{20}$$

subject to

$$\sum_{i=1}^{m} k_i = 1 \tag{21}$$

where

$$U_i(x) = \frac{x - worst}{best - worst} \tag{22}$$

The $>$ U is the winner!

Bibliography

[1] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag TELOS, Santa Clara, CA, USA, 3rd ed. edition, 2008.

[2] Thomas Bäck. *Evolutionary algorithms in theory and practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, New York, 1996.

[3] Michael Emmerich, Nicola Beume, and Boris Naujoks. An emo algorithm using the hypervolume measure as selection criterion. In *Evolutionary Multi-Criterion Optimization*, pages 62–76. Springer, 2005.

[4] Eckart Zitzler and Lothar Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *Evolutionary Computation, IEEE Transactions on*, 3(4):257–271, 1999.

[5] Padhraic Smyth. The em algorithm for gaussian mixtures. `http://www.ics.uci.edu/~smyth/courses/cs274/background_notes.html`, 2014. [Online; accessed 11-Oct-2014].

[6] David Barber. *Bayesian reasoning and machine learning*. Cambridge University Press, 2012.

[7] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977.

[8] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197, 2002.