Solving the Error Correcting Code Problem with Parallel Hybrid Heuristics

Lenguajes y Ciencias de la Computación

Enrique Alba and J. Francisco Chicano
Introduction

• Some applications cannot afford the resubmission of an erroneous msg

• Linear Block Codes

• Larger (Hamming) distance → Larger autocorrection capacity
ECC Problem

- Designing error correcting codes \((n, M, d)\) is NP-hard
- Objective: Given \(n\) and \(M\) find a code that maximizes \(d\)

Instance: \(n = 12\) bits, \(M = 24\) words

Maximum \(d\): 6
Genetic Algorithms

• Pseudo-code of a general GA

\[ t := 0; \]
\[ \text{initialize: } P(0) := \{\bar{a}_1(0), \ldots, \bar{a}_\mu(0)\} \in I^\mu; \]
\[ \text{evaluate: } P(0) : \{\Phi(\bar{a}_1(0)), \ldots, \Phi(\bar{a}_\mu(0))\}; \]
\[ \text{while } t(P(t)) \neq \text{true do} \quad \text{//Reproductive loop} \]
\[ \text{select: } P'(t) := s_{\Theta_s}(P(t)); \]
\[ \text{recombine: } P''(t) := \otimes_{\Theta_c}(P'(t)); \]
\[ \text{mutate: } P'''(t) := m_{\Theta_m}(P''(t)); \]
\[ \text{evaluate: } P''''(t) : \{\Phi(\bar{a}'''_1(t)), \ldots, \Phi(\bar{a}'''_\lambda(t))\}; \]
\[ \text{replace: } P(t + 1) := r_{\Theta_r}(P''''(t) \cup Q); \]
\[ t := t + 1; \]
\[ \text{end while} \]

• Decentralized GAs: cGA and dGA
Repulsion Algorithm

- RA considers the words as equally charged particles
- RA calculates the repulsion forces among the particles
- RA moves a particle according to the resultant force exerted over it

\[ f_{i,j} = \frac{1}{d_{i,j}} \frac{p_i - p_j}{\sqrt{d_{i,j}}} \]

\[ F_i = \sum_{j=1, j \neq i}^{M} f_{i,j} \]
Repulsion Algorithm

- RA considers the words as equally charged particles
- RA calculates the repulsion forces among the particles
- RA moves a particle according to the resultant force exerted over it

\[
F_i^n = (F_i \cdot \hat{n}_i)\hat{n}_i
\]

\[
F_i^t = F_i - F_i^n
\]

\[
F_i = \sum_{j=1, j \neq i}^{M} f_{ij}
\]
Repulsion Algorithm

- RA considers the words as equally charged particles
- RA calculates the repulsion forces among the particles
- RA moves a particle according to the resultant force exerted over it

\[
F^t_i = F_i - F^n_i
\]

\[
m^k_i = F^t_i \cdot e^k_i \geq \tau
\]
Hybrid Algorithms

- Hybridization: Inclusion of problem knowledge into the algorithm
- Two possible classes of hybrids:
  - **Strong**: Specific representation and operators
  - **Weak**: Combination of several algorithms (cooperation)

**GA main loop**
- Selection
- Recombination
- Mutation
- Replacement

**GARA main loop**
- Selection
- Recombination
- Repulsion
- Replacement
• De Jong’s function (possibly problematic):

\[ f(x) = \frac{1}{\sum_{i=1}^{M} \sum_{j=1, j \neq i}^{M} \frac{1}{d_{i,j}^2}} \]

• Corrected function

\[ f(x) = \frac{1}{\sum_{i=1}^{M} \sum_{j=1, j \neq i}^{M} \frac{1}{d_{i,j}^2}} + \left( \frac{d_{\text{min}}}{12} - \frac{d_{\text{min}}^2}{4} + \frac{d_{\text{min}}^3}{6} \right) \]
Fitness

- De Jong’s function (possibly problematic):
  \[ f(x) = \frac{1}{\sum_{i=1}^{M} \sum_{j=1, j \neq i}^{M} \frac{1}{d_{ij}^2}} \]

- Corrected function
  \[ f(x) = \frac{1}{\sum_{i=1}^{M} \sum_{j=1, j \neq i}^{M} \frac{1}{d_{ij}^2}} + \left( \frac{d_{min}}{12} - \frac{d_{2min}^2}{4} + \frac{d_{3min}^3}{6} \right) \]
Parameters

- Genotype: concatenation of the codewords
- The same number of evaluations for all the algorithms \((2 \times 10^5)\)
- Three distributed GAs: 5, 10 and 15 islands (same pop. size)
- Unidirectional ring for dGAs
- RA with \(\tau = 0.001\)
- Two classes of hybrids:
  - With recombination: PGARA\(_n\)
  - Without recombination: PGRA\(_n\)
Results

- The pure algorithms obtain a very small success rate (<4%)
- The fitness of solutions given by RA are lower than GAs ones
Results

- High success rate with hybrid algorithms
- Decentralized algorithms get still larger success than panmictic ones
- Results in 1 CPU better in both numerical effort and success
Results

- Higher hit rate in 5 CPUs
- Lower number of evaluations
- Improvement in the panmictic algorithm
Conclusions & Future Work

Conclusions

• Decentralization and hybridization with RA leverage the quality
• Increasing the number of islands do not always improves the search
• Results in 1 CPU better than 5 CPUs
• The recombination operator ruins the RA work

Future Work

• To study other algorithms for the problem (idea: scatter search + RA)
• To solve larger instances
• To extend the repulsion algorithm to other problems (Thomson)
Thanks for your attention !!!