优化和约束推理的动态分布式双向导遗传算法*

钟 静,应 宏
(重庆三峡学院 数学与计算机科学学院,重庆 万州 404000)

摘要:为了解决优化和约束推理,基于向导遗传算法(GGA)和分布式向导遗传算法(DGGA),通过引入向导概率 P_{guid}、本地优化监测 LOD 和权 ε 共 3 个新参数,提出了一种 D^3G^2A 算法的改进算法。该算法采用多代理方法,不仅使搜索过程多样化,避免出现局部最优,而且代理能计算各自的遗传参数。将改进的 D^3G^2A 和 GGA 用于随机生成的二元 CSPs,实验表明,D^3G^2A 能有效改善适应度值和节省 CPU 时间开销,使算法的性能得到提高。

关键词:优化;约束推理;动态分布式;双向导;遗传算法
中图分类号:TP183 **文献标识码**:A **文章编号**:1672-6693(2009)02-0094-05

约束满足问题(Constraint satisfaction problem,CSP)规则包括与论域和约束相关的变量及其子集。CSP 的解决方法就是将所有的变量按照各自论域中的值进行实例化,且必须满足所有的约束。方法的实现要花费很高的代价。为每个约束分配一个代价值,搜索所有变量的实例,以最小化违背约束的代价和。这个问题就是附加约束限制问题 CSPs,也称为 ∑CSPs[1]。但现实生活的问题更困难,必须优化目标函数以满足所有问题的约束。这就是约束满足与优化问题 CSOPs[2]。

∑CSPs 和 CSOPs 通常是 NP 问题,可以用完备的和不完备的方法来解决。完备的方法能解决优化问题,不完备的方法,如向导遗传算法 GGA 能避免局部最优[2];分布式向导遗传算法 DGGA[3],已成功应用在最大 CSP 中[4-5]。

1 ∑CSPs 和 CSOPs 规则

约束满足问题 CSP 是一个三元式(X,D,C),各组成部分定义如下:X 是变量 $\{x_1,x_2,\cdots,x_n\}$ 的有限集。D 是函数,将 X 中的每个变量在论域内映射成对应的值,可以是任意类型。D_{xi} 表示 x_i 经函数 D 映射后的目标集,$D=\{Dx_1,Dx_2,\cdots,Dx_n\}$。$C$ 表示约束的集合,每条约束限制了其所包含变量集赋值的组合,它是其所包含变量论域的笛卡儿乘积的一个子集,可为空。

CSOP 是一个四元式(X,D,C,f),其中(X,D,C)是 CSP 问题,f 是目标函数,它将每个实例映射成一个数值。

∑CSPs 可以定义成特殊的 CSPs,因为每个约束都有代价。目标函数 f 将每个实例映射成一个数值,表示违反约束的代价和。

2 动态分布式双向导遗传算法 D^3G^2A

2.1 基本原理

将初始种群分成子种群,每个子种群分配给称为种群代理的代理。种群包括具有相同范围适应度值的染色体,这个范围简称 FVR。种群代理能相互作用以优化解决问题。每个种群代理都按照各自的遗传算法工作,该遗传算法以模板理论[2]和最小冲突启发式[6]作为向导。在种群代理和用户之间有一个中间代理,当种群间互作用时,能监测到已达到的最好的解决方法。这个中间代理称为接口,它也可能建立新的种群代理。

2.2 最小冲突启发式和模板原理

每个染色体都附着在称为 $\text{template}_{i,j}$ 或 $\delta_{i,j}$ 的权建立起来的模板[2]。每个权对应 $\text{gene}_{i,j}$,i 代表染色体,j 表示位置。适应度模板用来定义不合要求的基因。

在 ∑CSPs 中,$\delta_{i,j}$ 表示 $\text{gene}_{i,j}$ 违反约束的代价和,而在 CSOPs 中,$\delta_{i,j}$ 表示应用在 $\text{gene}_{i,j}$ 上的处罚和。两者的权都是通过处罚算子更新,模板通过遗传算法完成替换。

* 收稿日期:2008-11-04
资助项目:重庆市教委科技计划(No. KJ081109);重庆三峡学院青年资助项目(No. 2006-sxxyqingnian-01)
作者简介:钟静,讲师,硕士,研究方向为神经网络、进化计算。

最小冲突启发式中运用的替换方法,见算法1。

算法1 与 chromosome_i 有关的最小冲突启发式算法。

```
Min-conflict-heuristic(chromosome)
    δ_{i,j}←max(template_i) /* δ_{i,j} is associated to gene_{i,j} which is in turn associated to the variable v_j */
    FV* ←0
  for each value in domain of v_j do
    {FV* ←compute-fitness-value(value)
    if (FV > FV*) then
      FV* ←FV
    end if
      value* ←value
      value(gene_{i,j}) ←value*
      update(template_i)
     }
  return FV*
```

2.3 适应度函数

遗传算法的核心是适应度函数。∑CSPs 和 CSOPs 使用的适应度函数不一样。

在∑CSPs(X,D,C,f)中,目标函数可以定义成 $f(ps) = \sum WC_i$。其中,C_i 是违反约束,WC_i 是代价。在遗传算法中总是取染色体的最大适应度值。因此,适应度函数可定义为 $g(ps) = \text{Sum\_tc} - f(ps)$。其中,Sum_tc 表示所有约束的代价和。

在 CSOPs 中,定义一个扩大的适应度函数 g,用于优化。$g(ps) = f(ps) + \lambda * \sum(PC_i * I_i(ps))$。函数 g 表示一个染色体的目标函数和以及存在的处罚,ps 表示潜在的方法,λ 是正则化参数,决定适应度函数值中的处罚的分配比率,$I_i(ps)$ 是方法的指示器,如果 ps 满足所有的约束它就等于1,否则为0。

2.4 代理结构

每个代理都有一个简单的结构:认知(代理知道该和谁通信),本地知识(包括静态和动态知识),邮箱(存储和接收信息,用于以后逐一处理)。

1)种群代理。种群代理熟悉其它种群代理和接口代理。静态知识包含∑CSP 或 CSOP 的数据(如变量、域值、约束)、特殊性(如适应度函数范围)、本地遗传参数(变异概率、交叉概率、代数等)。动态知识将组件作为种群池,从一代向另一代变化(包括染色体、种群大小)。

2)接口代理。接口代理对所有的种群代理都很熟悉。静态知识包括∑CSP 或 CSOP 的数据。动态知识包括最优染色体。

2.5 全局动态

接口代理随机产生初始种群,根据种群特点(如适应度范围 FVR)分子种群。初始种群建立种群代理,以分配相应的子种群。接口代理请求子种群优化处理。在开始自身优化前,每个种群代理、种群 FVR,初始化所有的模板、对应染色体的处罚数。然后初始子种群执行遗传过程,返回子种群'pop',它只完成交叉和变异步骤一次。对 pop 中的每个染色体,种群 FVR 计算适应度函数值 FV。可能出现两种情况:一种情况对应与父辈具有相同范围 FV 的染色体。这种染色体替换以后随机选择的其中一个染色体;第二种情况就是 FV 不在相同的范围 FVR 中。如果这种代理已经存在,染色体就要发送给另一个种群 FV,否则发送给接口代理。接口代理建立一个具有 FV 的新代理,把上述染色体发给它。无论何时建立新的种群代理,接口代理都会通知其它代理,并请求新种群执行优化。信息的处理具有优先权,因此,无论何时代理接收到信息,都会停止自身行为,保存信息,更新本地知识,在重新工作前保存信息。

代理的行为描述了遗传子过程,见算法2。每个种群代理将它的行为应用到子种群,直到满足迭代(代数)。

算法2 与 $\text{Specie}_{\text{FVR}}$ 有关的行为。

```
Apply-behavior (initial-population)
    init-local-knowledge
    for k := 1 to number-of-generations do
        {template-updating (initial-population)
        pop←genetic-process (initial-population)
        best-FV←0
        for each chromosome_j in pop do
         FV_j←compute-fitness-value (chromosome)
         if best-FV ≤ FV_j then
         best-FV←FV_j
              clear (LO-chromosomes-list)
               LO-chromosomes-list ← LO-chromosomes-list ∪ {chromosome_j}
        end if
        if (FV_j ∈ range_i) then
         replace-by (chromosome_j)
    else if exist-agent (Species_FV) then
              sendMsg (Species_i, Species_FV, 'take-into-account (chromosome_j)')
              else sendMsg (Species_i, Interface, 'creat-agent (chromosome_j)')
```

```
        end if
    end if
    end for
    if best-FV = last-FV then
        stat-counter←stat-counter + 1
        Else last-FV←best-FV
    end if
    if stat-count = LOD_i then
        last-stat-FV←last-FV
            Penalize (LO-chromosomes-list)
        sendMsg (Species_i, Interface, ' result (one-chromosome, specificity) ')
    end if
    }
```

2.6 向导交叉与向导变异

每对染色体外,交叉算子产生一个新的子代,其描述如算法 3 ~ 4。子代继承了最好的基因。

算法 3　交叉算子。

```
Cross-over (chromosome_i1, chromosome_i2)
    for j := 1 to size (chromosome_i1) do
        {sum←template_i1,j + template_i2,j
    if (random-integer[0,sum-1] < template_i1,j) then
        gene_i3,j←gene_i2,j
        else gene_i3,j←gene_i1,j
    end if
    }
    return chromosome_i,3
```

算法 4　与 $Specie_{FVR}$ 有关的交叉过程。

```
Crossing (mating-pool)
    if (mating-pool size < 2) then
      return mating-pool
    end if
    for each pair in mating-pool do
      if (random[0,1] < P_cross) then
       offspring←cross-over (first-pair, second-pair)
        FV←compute-fitness-value (offspring)
    offspring-pool←offspring-pool ∪ {offspring}
      end if
    return offspring-pool
```

2.7 处罚算子和本地优化监测

在 D^3G^2A 中,增加了另一个遗传算法参数 LOD (Local optima detector,本地优化监测)。它表示邻居没有改进时的代数。例如,对于特定代数,如果最优染色体的 *FV* 保持不变,可以断定代理优化子过程陷入了局部最优。事实上,如果未变的 *FV* 比最后固定的 *FV* 小,LOD 就会自动为 1,见算法 9。否则 LOD 保持不变。LOD 是整个优化过程中的参数,它会被每个代理动态更新。

每个种群 FVR 都会为下一代保存最好的 *FV*。这不仅对于固定的适应度值有用,也有助于在种群中选择最优染色体。如果最好的 *FV* 对于 LOD 代保持不变,就视为陷入局部最优。具有 *FV* 的染色体,所有基因的相关模板会增加 1,染色体模板将用处罚来更新。

2.8 变异与向导概率

传统的遗传算法中,变异的目的是为了种群多样化。如果基因值在初始种群中不存在,交叉过程中就无法获取。因此,为了生成可能丢失的基因值,有时必须要随机变异。

D^3G^2A 中使用了一个随机操作数,即向导概率 P_{guid}。变异子过程发生了变化 按照变异概率 P_{mut} 选择的每个染色体,根据概率 $1 \sim P_{guid}$ 随机变异,概率 P_{guid} 作为向导(见算法 5 的第 2 句)。变异子过程见算法 5 ~ 7。

算法 5　与 $Specie_{FVR}$ 有关的变异过程。

```
Mutating (offspring-pool)
    for each chromosome in offspring-pool do
      {if (random[0,1] < P_muti) then
     if (random[0,1] < P_guid) then
        Guided_mutation (chromosome_i)
       end if
        else Random_mutation (chromosome_i)
     end if
     FV* ←compute-augmented-fitness-value (chromosome_i)
     Offspring-pool-mutated←offspring-pool-mutated ∪ {chromosome_i}
    }
    return offspring-pool-mutated
```

算法 6　与 $chromosome_i$ 有关的随机变异。

```
Random_mutation (chromosome_i)
    choose randomly a gene_i,j
    choose randomly a value v_i in domain of gene_i,j
    value (gene_i,j) ←v_i
    return chromosome_i
```

算法 7　与 $chromosome_i$ 有关的向导变异。

```
Guided_Mutation (chromosome_i)
    Min-conflict-heuristic (chromosome_i)
    return chromosome_i
```

2.9 动态方法

D^3G^2A 中第二个改进是基于达尔文理论和自然规则。在动物界,强壮的成员有幸繁殖,并能避免疾病的感染。相反,弱者频繁生病,并不能吸引异性。强的

种群间交叉,弱的种群发生变异,这是很有价值的。

因此 P_{cross} 和 P_{mut} 将影响适应度,构成一个新的算子 ε,这个操作数就是权,其范围在 0 ~ 1 间。在染色体适应度值基础上,种群代理根据 ε 来决定自身遗传过程的参数。

在优化过程中,如算法 8 ~ 9,每个种群代理按自己的遗传过程进行。事实上,在开始优化过程前,代理必须基于适应度值计算参数 P_{cross} 和 P_{mut}。给出的种群代理在新的遗传过程中会出现 3 种可能情况。

算法 8 遗传过程。

```
Genetic process
    mating-pool←matching (population-pool)
    (P_crossi, P_muti, LOD_i) ←count_operator (P_cross, P_mut, LOD)
    template-updating (mating-pool)
    offspring-pool-mutated←mutating (offspring-pool-crossed)
    return offspring-pool-mutated
```

算法 9 算子计算过程。

```
Count_operator(P_cross, P_mut, LOD)
    if FV < (max-attained-FV/2) then
        P_crossi ← P_cross / ε
        P_muti ← P_mut t * ε
        LOD_i ← LOD/ε
    end if
    if FV > (max-attained-FV/2) then
        P_crossi ← P_cross * ε
        P_muti ← P_mut / ε
        LOD_i ← LOD * ε
    end if
    if FV = (max-attained-FV/2) then
        P_crossi ← P_cross
        P_muti ← P_mut
        LOD_i ← LOD
    end if
    if FV ≤ last-stat-FV then
        LOD_i ← 1
    end if
    return (P_crossi, P_muti, LOD_i)
```

3 实验结果

使用随机二元约束满足问题测试 D^3G^2A。随机约束满足问题有 4 个参数:n 表示变量的个数,d 表示变量论域的大小,p 表示约束网络中约束占所有可能约束的比例,q 表示约束网络中满足约束关系的值对个数占所有可能值对的比例。实验中设 $n = 20, d = 20, p, q$ 选择如下的值:0.1,0.3,0.5,0.7,0.9,从而获得了 25 个 p-q 的组合。对于每种组合,随机生成 30 个样本,最终获得了 750 个样本。随机代价已分配给了每个约束。考虑到遗传算法的随机性,对每个样本作了 10 次实验,不考虑异常取平均值。对每种 p-q 组合,也取 30 个样本的平均值。

关于遗传参数,所有的实验中,设代数 NG 为 10,初始种群大小为 1 000,交叉概率 0.5,变异概率 0.2,和一个随机替换。LOD 的初值为 3,ε 为 0.6。实验中的参数值是在最佳 CPU 时间和最好适应度值时给出的。

性能评价:运行时间(CPU 用来解决问题花费的时间)说明复杂性;满足度(满足约束的数目)保证质量。

实验分别在向导遗传算法(GGA)和 D^3G^2A 两种方法下完成。其性能比较如下:对于相同适应度值,计算每种方法使用的时间;对于相同 CPU 时间,计算每种方法的适应度值。

测试结果如图 1 ~ 2 所示。从 CPU 时间来看(图 1),D^3G^2A 方法需要较少的或相同的时间。特别对于过度约束的样本集来说,D^3G^2A 需要更少的时间。从解决方法的质量看(图 2),D^3G^2A 总能比 GGA 找到更好的方法。对于更多约束的问题而言,D^3G^2A 是 GGA 的 3 倍多。这就是 D^3G^2A 方法中使用了搜寻多样性和参数增强性的结果。

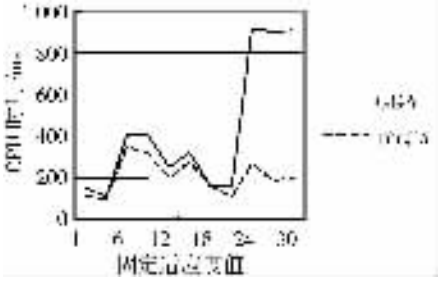

图 1 给定适应度值对应的 CPU 时间

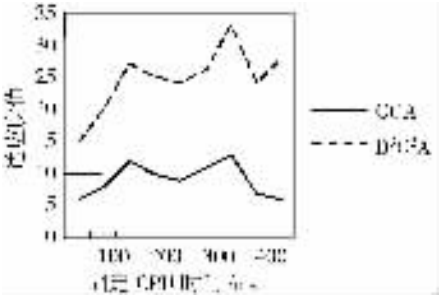

图 2 给定 CPU 时间对应的适应度值

4 结论

针对约束满足和优化问题,提出了一种新的方

法 动态分布式双向导遗传算法(D^3G^2A)。该方法是在标准遗传算法[7-8]中引入了向导概率 P_{guid},本地优化监测 LOD 和权 ε 共 3 个新的参数,提高了算法的性能。实验表明,D^3G^2A 能有效改善适应度值和节省 CPU 时间开销。事实上,问题越复杂,这个方法的解决办法越好。

当然 D^3G^2A 方法的性能还需进一步提高,今后的工作主要集中在用该方法来解决现实生活中复杂的问题以及分布式 CSPs。

参考文献:

[1] 杨轻云,孙吉贵,张居阳. 最大度二元约束满足问题粒子群算法[J]. 计算机研究与发展,2006,43(3):436-441.

[2] Tsang E P K,Wang C J,Davenport A,et al. A family of stochastic methods for constraint satisfaction and optim-ization [R]. Colchester:Technical Report University of Essex,1999.

[3] Ghedira K, Jlifi B. A distributed guided genetic algorithm for max_CSPs[J]. Journal of Sciences and Technologies of Information (RSTI),Journal of Artificial Intelligence Series (RIA),2002,16(3):192-207.

[4] Bouamama S,Ghedira K. ED^3G^2A:an enhanced version of the dynamic distribute double guided genetic algorithms for max_CSPs[C]. In proceedings of the 8th world multi- conference on systemics, Cybernetics and Informatics (SCI'04),Orlando,Florida,USA,2004.

[5] Bouamama S,Ghedira K. D^2G^2A and D^3G^2A:a new generation of distributed guided genetic algorithms for max_CSPs [C]. In proceedings of the 7th world multiconference on systemics,Cybernetics and informatics(SCI 03), Orlando, Florida,USA,2003.

[6] Lau T L,Tsang E P K. Solving the radio link frequency assignment problem with the guided genetic algorithm[C]. In proceedings of NATO symposium on radio length frequency assignment staring and conservation systems, Albrg,Denmark,UK,1998.

[7] 杨晓琪. 约束最优化问题的非线性无约束方法[J]. 重庆师范大学学报(自然科学版),2004,21(2):1-3.

[8] 刘双印. 改进小生境遗传算法在元搜索引擎调度优化中的研究[J]. 重庆师范大学学报(自然科学版),2008,25(3):46-50.

A Dynamic Distributed Double Guided Genetic Algorithm for Optimization and Constraint Reasoning

ZHONG Jing, YING Hong

(College of Mathematics and Computer Science, Chongqing Three Gorges University, Chongqing 404000, China)

Abstract: A Dynamic Distributed Double Guided Genetic Algorithm (D^3G^2A) is a new multi-agent approach which leads to additive constraint satisfaction problem. This approach is inspired by the guided genetic algorithm(GGA) and by the dynamic distributed double guided genetic algorithm for Max_CSPs. It consists of agents dynamically created and cooperating in order to solve problem with each agent performing its own GA. Firstly, our approach is enhanced by three parameters, guidance probability (P_{guid}), local optima detector (LOD), weight (ε), which allow not only diversification but also escaping from local optima. Secondly, the GGAs performed agents will no longer be the same. This is stirred by the natural laws. In fact, our approach will let the agents able to count their own GA parameters. In order to show D^3G^2A advantages, the approach and the GGA are applied to the randomly generated binary constraints satisfaction problems. Compared with the centralized guided genetic algorithm and applied to a set of literature known problems, our new approaches have been experimentally shown to be always better in terms of fitness values and CPU time.

Key words: optimization; constraint reasoning; dynamic distribution; double guided; genetic algorithm

(责任编辑 游中胜)