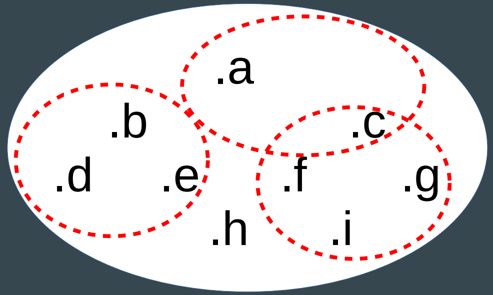
Otimização Combinatória Discreta e Programação Linear Inteira
...

Problema de otimização combinatória discreta (OC) – o que é?



...

- I = conjunto de elementos
- C = coleção de subconjuntos dos elementos de I
- f(S) = valor associado a cada subconjunto S em C

Qual o conjunto S em C que minimiza/maximiza f(S) ?

Programação Linear Inteira (PLI) – o que é?

A área de PLI reúne técnicas, algoritmos e teorias em torno da classe de problemas de OC em que a coleção C e a função de otimização f podem ser expressas por equações lineares.

Exemplos:

Montagem de tabelas de horários: aulas em escolas, viagens de ônibus, etc.
Montagem de escalas de trabalhos: enfermarias de hospitais, tripulação de aviões, etc.
Planejamento de produção: sequenciamento de máquinas, controle de estoques, etc.
Telecomunicações: localização de antenas de celulares, planejamento de expansão de redes de telefonia, etc.
Roteamento: logística de distribuição, caminhos mais curtos, etc.
Projetos de circuitos integrados,
etc.

Edna A. Hoshino

CV: http://lattes.cnpq.br/9160943574918408

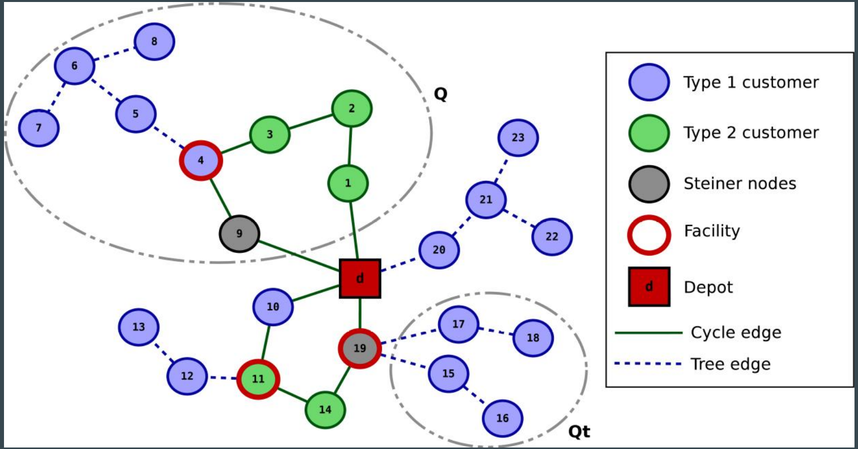
Alguns problemas estudados:

- PHS: problema do horário e alocação de salas de aulas da Facom
- Filogenia Viva
- Problemas de Seleção de Strings - M
- Heurísticas para problemas de roteamento em larga escala
- Matheurísticas para o problema da pseudoarborescência orientada a lucro
- Problema das pseudo-arborescências capacitadas com localização de facilidades - M
- Problema de roteamento em anéis de dois níveis (ring/ring) - M
- Coloração de vértices com pesos dissonantes e restrições de cores*- M
- Matheurísticas para o problema do horário e alocação de salas da Facom*

Projetos de pesquisa em andamento:

- Programação linear inteira aplicada a problemas de biologia computacional
- Algoritmos exatos para variantes do problema dos m-anéis-estrelas capacitados

Algoritmos exatos para variantes do problema dos m-anéis-estrelas capacitados



Este projeto tem por objetivo propor algoritmos exatos baseados no método da geração de colunas para resolver variantes do CmRSP. Neste tipo de problema, clientes estão geograficamente dispersos e demandam produtos ou serviços disponíveis em um depósito central. A distribuição dos produtos ou serviços é realizada em dois níveis por uma frota limitada de veículos idênticos com uma capacidade de carga definida. O primeiro nível de distribuição é custoso e consiste em um ciclo passando pelo depósito. O segundo nível é realizado por um meio mais barato, por exemplo com árvores. Cada veículo realiza a distribuição de produtos e serviços usando um dos dois níveis de distribuição ou uma combinação de ambos. Neste tipo de problema, clientes classificados como prioritários devem ser servidos somente pelo primeiro nível. O problema consiste em encontrar a melhor distribuição dos produtos e serviços.

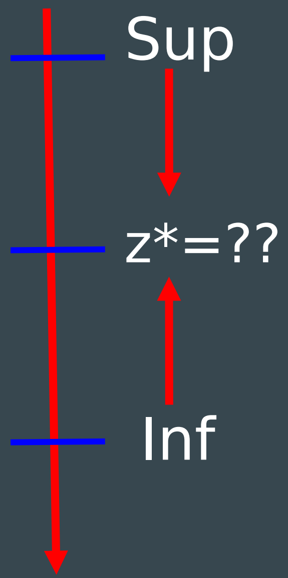
Este tipo de problema tem aplicações em situações práticas como na área de telecomunicações, projeto de redes de comunicação, logística de distribuição, etc.

Técnicas usadas: Programação Linear Inteira, Geração de Colunas, Branch-and-cut-and-price

As seguintes variantes foram estudadas: CPAFLP, PRAP e Ring/Ring.

Técnicas:

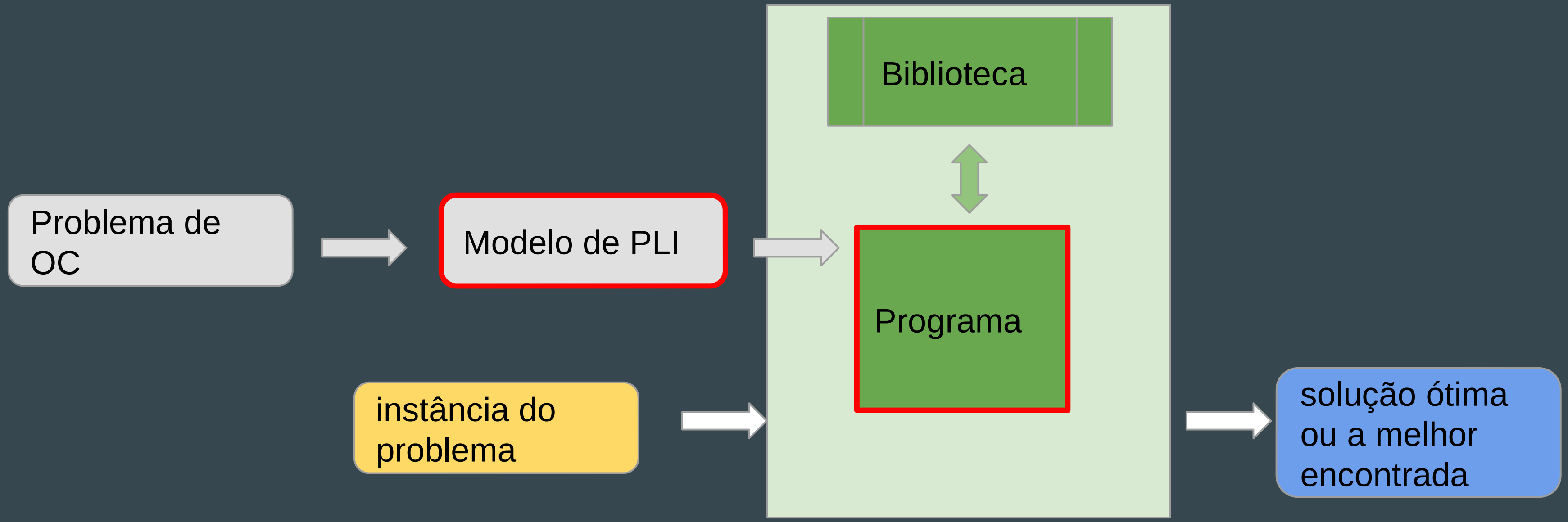
Programação Linear Inteira (PLI)
Geração de Colunas (GC)
Planos de Cortes (CP)
Branch-and-Bound (BB)
Branch-and-cut-and-price = BB + CP + GC
Matheuristicas = metaheurísticas + modelagem matemática



Bibliotecas/linguagens/ambientes:

Resolvedores: SCIP, CPLEX, XPRESS
Linguagem de programação: C/C++
Sistema Operacional: Linux
Laboratório: Lexa

Metodologia



Exemplo de Modelagem matemática por PLI

$$
\begin{array}{llllr}
\text{(SPF)} & \min \sum_{p \in P} c_p \lambda_p & & & \\
\text{sujeito a} & \sum_{p \in P} \lambda_p \leq m & & (\pi) & (4.1) \\
& \sum_{p \in P} (r_i^p + t_i^p) \lambda_p = 1 & \forall i \in U_1 & (\alpha) & (4.2) \\
& \sum_{p \in P} r_i^p \lambda_p = 1 & \forall i \in U_2 & (\mu) & (4.3) \\
& \sum_{p \in P} (r_i^p + t_i^p) \lambda_p \leq 1 & \forall i \in W & (\gamma) & (4.4) \\
& \sum_{p \in P} (r_{ij}^p + t_{ij}^p) \lambda_p \leq 1 & \forall ij \in A & (\beta) & (4.5) \\
& 0 \leq \lambda_p \leq 1 & \forall p \in P. & & (4.6)
\end{array}
$$

Exemplo de resultados*

Instâncias	Com *GAP* MCF/SPF(=)	*GAP* MCF	*GAP* SPF	Ótimos MCF/SPF(=)	Tempo MCF	Tempo SPF
CF0_QT1	45 / 16 (16)	50,16	4,55	0 / 4 (0)	*	*
CF0_QT66	45 / 17 (17)	40,86	3,16	1 / 8 (1)	1167,2	161,7
CFMED_QT1	45 / 17 (17)	59,84	3,12	0 / 4 (0)	*	*
CFMED_QT66	45 / 14 (14)	26,59	1,39	2 / 7 (1)	1706,0	173,0
Soma	180 / 64 (64)			3 / 23 (2)		
Média		44,36	3,06		1436,6	167,4

Tabela 5.18: Desempenho MCF $\times$ SPF.

* dissertação do Fábio Abe

Exemplo de programa*

emacs@ednaduo

File Edit Options Buffers Tools C Help

Save Undo

```
*/
SCIP_RETCODE SCIPprobdataCreate(
   SCIP*                   scip,              /**< SCIP data structure */
   const char*             probname,          /**< problem name */
   instanceT*              I                  /**< instance of knapsack */
   )
{
   SCIP_PROBDATA* probdata;
   SCIP_CONS** conss;
   SCIP_VAR** vars, * var;

   char name[SCIP_MAXSTRLEN];
   int i;
   int ncons;
   int nvars;

   assert(scip != NULL);

   /* create problem in SCIP and add non-NULL callbacks via setter functions */
   SCIP_CALL( SCIPcreateProbBasic(scip, probname) );

   SCIP_CALL( SCIPsetProbDelorig(scip, probdelorigMochila) );
   SCIP_CALL( SCIPsetProbTrans(scip, probtransMochila) );
   SCIP_CALL( SCIPsetProbDeltrans(scip, probdeltransMochila) );
   SCIP_CALL( SCIPsetProbInitsol(scip, probinitsolMochila) );
   SCIP_CALL( SCIPsetProbExitsol(scip, probexitsolMochila) );

   /* set objective sense */
   SCIP_CALL( SCIPsetObjsense(scip, SCIP_OBJSENSE_MAXIMIZE) );

   /* TODO: tell SCIP that the objective will be always integral (it depends on
the problem) */
```