# Using DCOP to Model Resource Allocation: A Review of Algorithms

Alexander R. Gularte, Diana Francisca Adamatti
Programa de Pós Graduação em Computação
Centro de Ciências Computacionais
Universidade Federal do Rio Grande (FURG)
Rio Grande – RS – Brasil

***Abstract*—Distributed Constraint Optimization Problem (DCOP) is a formalism that is widely used for coordination in multiagent systems. The advantage of applying these algorithms for multiagent coordination is due to the fact that them are distributed, robust and scalable. The aim of this work is to present a revision of the complete and incomplete algorithms, generally found in the literature and how these approaches can benefit the resource allocation in Multiagent Systems.**

## I. Introdução

Diversos formalismos podem ser utilizados para coordenação de Sistemas Multiagente (SMA), contudo os Problemas de Otimização de Restrição Distribuída (DCOP - *Distributed Constraint Optimization Problems*) vêm ganhando destaque na literatura por ser uma abordagem robusta e escalável. Os DCOPs estão associados a outros três formalismos: Problemas de Satisfação de Restrição (CSP - *Constraint Satisfaction Problem*)[Apt 2003], Problemas de Otimização de Restrição (COP - *Constraint Optimization Problems*)[Schiex et al. 1995] e Problemas de Satisfação de Restrição Distribuída (DCSP - *Distributed Constraint Satisfaction Problems*)[Yokoo et al. 1992].

Um CSP consiste em associar valores a variáveis, de tal forma que um conjunto de restrições é satisfeito. Cada uma das *n* variáveis do problema ($X_1$,$X_2$,...,$X_n$) assume um valor que pertence a um determinado domínio discreto. Um conjunto de domínios ($D_1$,$D_2$,...,$D_n$) especifica o domínio de cada uma das variáveis. Uma restrição é definida por um predicado. Uma restrição $P_k$ = ($xk_1$,...$xk_j$) é um predicado definido sobre o produto cartesiano $Dk_1 \times \ldots \times Dk_j$. O predicado será verdadeiro se os valores associados às variáveis satisfazem a restrição. A solução para um CSP é equivalente a encontrar uma associação de valores para cada uma das variáveis envolvidas no problema de tal forma que todas as restrições sejam satisfeitas.

A extensão dos CSPs para coordenar a resolução cooperativa de problemas distribuídos em SMA foi proposta por [Yokoo et al. 1992]. Para tornar isso possível, as variáveis do CSP foram distribuídas entre os agentes. Essa abordagem pôde ser aplicada, por exemplo, na alocação de tarefas em SMA, considerando cada tarefa como uma variável e os domínios das variáveis como o conjunto de agentes capazes realizá-las.

O trabalho de [Schiex et al. 1995] propôs generalizar as funções booleanas dos CSPs em funções valoradas. Esses valores denotam o impacto causado pela violação da restrição. Além disso, permitem representar níveis de preferência em relação às possíveis associações de valores. O processo de resolução de um COP tende a ser mais complexo que um CSP, pois, não basta encontrar uma associação que satisfaz todas as restrições, é necessário encontrar a associação que otimiza o valor das funções de custo. Isso implica em uma maior exploração do espaço de estados.

Baseando-se nos formalismos mostrados anteriormente, diversos pesquisadores propuseram os DCOPs, que podem ser vistos tanto como uma distribuição dos COPs, como uma generalização dos DCSPs. Um DCOP é formalmente definido como a tupla ($X$ , $D$ , $C$ , $A$, $\alpha$), onde $X = \{x_1,x_2,\ldots,x_n\}$ é um conjunto de $n$ variáveis, $D = \{D(x_1),D(x_2),...,D(x_n)\}$ um conjunto de domínios discretos no qual cada elemento corresponde ao domínio de uma variável, $C$ um conjunto de funções de custo, $A$ o conjunto de agentes e $\alpha$ o mapeamento de agentes e variáveis. Encontrar uma solução de custo mínimo para um DCOP é um problema NP-Hard [Modi et al. 2005].

Em [Yeoh 2010] foi descrita uma possível taxonomia para abordagens de solução para DCOPs presentes na literatura. Neste artigo é apresentada uma extensão da taxonomia de Yeoh onde os algoritmos incompletos são expandidos nos que tem garantia de qualidade e nos que não tem. A taxonomia estendida é mostrada na Figura 1.

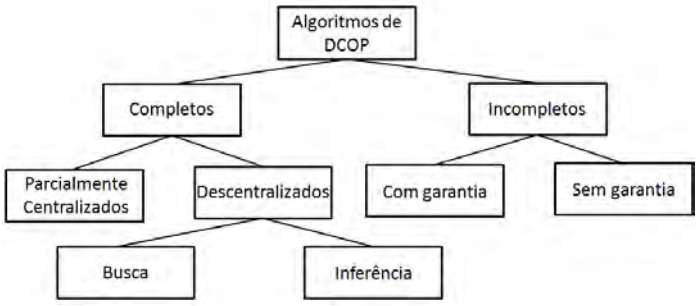


Fig. 1. Taxonomia dos algoritmos para solucionar DCOPs.

Inicialmente, os algoritmos podem ser divididos em completos e incompletos. Os completos fornecem soluções ótimas ao custo de muita computação e trocas de mensagens. Por outro lado, os incompletos encontram soluções sem garantia de qualidade mais rapidamente.

Na classe dos algoritmos completos, os parcialmente centralizados compreendem os algoritmos nos quais alguns agentes transferem as suas informações de restrição (funções

de custo) para que outros agentes centralizadores realizem o processamento. O algoritmo mais conhecido dessa classe é o OptAPO (*Optimal Asynchronous Partial Overlay*) [Mailler e Lesser 2004]. O OptAPO apresenta uma abordagem baseada na cooperação de mediadores, onde alguns agentes atuam como mediadores resolvendo centralmente subproblemas que se sobrepõem.

Diferentemente dos parcialmente centralizados, nos descentralizados cada agente tem acesso apenas as suas restrições. Essa classe se divide em uma abordagem baseada em busca e outra em inferência. Os de inferência geralmente utilizam programação dinâmica para propagar as funções de custos de um agente para outro. O algoritmo mais referenciado na literatura que utiliza essa abordagem é o DPOP (*Distributed Pseudo-tree Optimization*) [Petcu e Faltings 2005]. Já os algoritmos baseados em busca empregam estratégias de busca distribuída para explorar o espaço de soluções até encontrar aquela que possui o custo mínimo.

A classe dos algoritmos incompletos tende a ter maior aplicação em problemas reais em virtude do reduzido custo computacional. Apesar de não possuírem garantia de encontrar a solução ótima, alguns fornecem limites de distância máxima entre a solução ótima e a aquela que pode ser encontrada. Os algoritmos sem garantia tendem a ser os mais rápidos para solucionar DCOPs, contudo não apresentam qualquer garantia em termos de convergência e qualidade da solução.

## II. Algoritmos Completos

### A. ADOPT - Asynchronous Distributed Constraint Optimization

O algoritmo ADOPT (*Asynchronous Distributed Optimization*), proposto por [Modi et al. 2005], se destaca como o primeiro método completo, distribuído, assíncrono e com comunicação local. Essa última característica demonstra a escalabilidade do método, pois um agente se comunica apenas com seus vizinhos. Dois agentes são considerados vizinhos sempre que existir ao menos uma função de custo entre eles. No ADOPT, a assincronia é alcançada à medida que os agentes podem mudar o valor de suas variáveis sempre que percebam a existência de uma solução melhor que a atual. Além disso, essas decisões são realizadas baseando-se em informações locais. A busca assíncrona no ADOPT é uma variante da busca em profundidade *Branch-and-Bound* (BB).

### B. DPOP - Distributed Pseudo-tree Optimization

Diferentemente do ADOPT, o algoritmo DPOP (*Dynamic Programming Optimization Protocol for* DCOP) proposto em [Petcu e Faltings 2005] utiliza uma estratégia de inferência baseada em programação dinâmica. A principal vantagem do DPOP é utilizar um número linear de mensagens, o que reduz o *overhead* dos algoritmos de busca distribuída. A complexidade do DPOP depende do tamanho de suas mensagens de utilidade, que são limitadas exponencialmente.

O algoritmo DPOP é composto por três fases: criação da árvore DFS (*Depth First Search*), propagação das mensagens de utilidade e propagação das mensagens de valor. A primeira fase consiste em gerar a árvore DFS para o grafo de restrições do DCOP. Em seguida, ocorre um processo de propagação de utilidades que começa nas folhas e vai até a raiz. A ideia dessa fase é ir coletando informações de utilidade a fim de fornecer subsídios para os nós superiores escolherem os valores que proporcionam maior ganho de utilidade. Terminada essa fase, inicia-se um processo de propagação de valores, onde, desde a raiz, os nós começam a associar os valores ótimos a suas variáveis.

### C. OptAPO - Optimal Asynchronous Partial Overlay

A maioria dos algoritmos para solução distribuída de DCOPs mantem uma separação total entre os conhecimentos dos agentes ao longo do processo de resolução. Contudo, em muitos casos, se os agentes pudessem compartilhar as suas funções de custo, soluções estáveis poderiam ser encontradas mais rapidamente. Motivado por esse inconveniente, [Mailler e Lesser 2004] apresenta um novo algoritmo baseado em uma técnica parcialmente centralizada denominada mediação cooperativa. Assim, o OptAPO fornece uma resolução rápida, distribuída e assíncrona sem gerar o *overhead* decorrente das comunicações excessivas. O OptAPO se propõe a explorar a velocidade das técnicas centralizadas, ao mesmo tempo que de forma distribuída consegue identificar as subestruturas do problema.

## III. Algoritmos Incompletos

### A. Algoritmo Soma-Máxima

Uma das formas de representar um DCOP é através de grafos bipartidos denominados grafos-fatores. Um grafo bipartido é composto por arestas não direcionais e dois conjuntos de nós. Nesses grafos, cada aresta conecta nós de conjuntos diferentes. No caso dos grafos-fatores, um conjunto de nós representa as variáveis das funções (nós de variável), enquanto os outros nós representam as funções (nós de funções). As arestas conectam as variáveis às funções sempre que uma variável for argumento para uma função.

[Farinelli et al. 2008] propôs representar um DCOP através de grafos-fatores e então aplicar o algoritmo soma-máxima, extensão do soma-produto [Kschischang et al. 2001], para encontrar soluções aproximadas através de troca de mensagens locais. Quando aplicado em um grafo-fator acíclico, o soma-máxima tem garantia de encontrar a solução ótima, por outro lado em grafos cíclicos essa garantia não existe. O soma-máxima tem a vantagem de apresentar ótima escalabilidade, uma vez que a complexidade no cálculo das mensagens depende apenas do número de vizinhos e não do número total de agentes.

O algoritmo soma-máxima opera através da troca de mensagens enviadas de nós de função para nós de variável e de nós de variável para nós de função. Essas mensagens tratam-se de funções de custo que ao longo da execução do algoritmo são somadas e margilizadas. Essas funções resumem todo custo existente na porção do grafo da qual elas provêm. O algoritmo termina quando todos os nós de variável tenham recebido todas as mensagens de seus vizinhos. Nesse momento, cada nó de variável terá subsidios para decidir qual melhor valor para assumir, uma vez que irá conhecer o custo existente em todas as direções ao seu redor.

### B. Algoritmo Soma-Máxima Branch-and-Bound

O trabalho de [Stranders et al. 2009] identificou um gargalo em potencial nas mensagens que são enviadas das funções para as variáveis no soma-máxima. Para gerar essas mensagens é necessário produzir todas as possíveis combinações de valores para as variáveis. Contudo, o espaço de estados em questão cresce exponencialmente em relação ao número de variáveis e o número de possíveis valores/estados de cada variável. Com base nesse inconveniente, [Stranders et al. 2009] propõe dois algoritmos de poda. O primeiro deles é executado em uma etapa de pré-processamento, antes de executar o soma-máxima propriamente dito, e baseia-se na poda de estados dominados. Esses estados consistem em valores atribuídos às variáveis que comprovadamente não levam a uma solução ótima global. O segundo algoritmo emprega a técnica de busca *Branch-and-Bound* para otimizar a exploração de estados necessária ao gerar as mensagens de funções para variáveis. Estudos posteriores referenciam esses dois algoritmos como uma evolução do soma-máxima, que pode ser chamado de Soma-Máxima *Branch-and-Bound* [Macarthur 2011].

### C. Algoritmo Soma-Máxima Limitado

O trabalho de [Rogers et al. 2011] propõe uma abordagem que fornece um limite entre a solução encontrada pelo soma-máxima e a ótima. Essa evolução foi chamada soma-máxima limitado (*bounded max-sum*). A ideia básica do soma-máxima limitado é remover os ciclos do grafo-fator para obter resultados ótimos com o algoritmo soma-máxima. Entretanto, para remover ciclos é necessário ignorar algumas dependências entre funções e variáveis (arestas do grafo-fator), o que acaba gerando um segundo problema, diferente do inicial representado pelo grafo cíclico. Não existe garantia de que a solução ótima para o grafo livre de ciclos será também uma solução ótima para o grafo cíclico. Contudo é possível estabelecer uma distância limite entre a solução ótima do grafo cíclico e aquela encontrada através do grafo acíclico. Uma etapa chave dessa abordagem é escolher as dependências que serão removidas, considerando o impacto que elas terão na qualidade da solução.

## IV. Modelagem de Alocação de Tarefas com DCOP

Alocação de tarefas envolve associar um conjunto de tarefas a um conjunto de agentes, onde ambos variam com o tempo, ou seja, o ambiente é dinâmico. Cada agente recebe uma recompensa baseada em uma função de utilidade que determina o ganho para cada associação agente-tarefa. Assim, uma solução ótima será a associação cujo ganho total para todos os agentes seja maximizado.

Dentre as possíveis soluções para o problema de alocação de tarefas, os DCOPs se destacam por serem uma representação flexível o suficiente para representar as mudanças rápidas no ambiente sem a necessidade de criar uma representação completa do mesmo. [Macarthur 2011] utiliza essa abordagem através de grafos-fatores, onde os agentes são representados por nós de variável e as tarefas por nós de função. As principais contribuições do trabalho de Macarthur são três algoritmos para alocação dinâmica de tarefas: *fast-max-sum*, *branch-and-bound fast-max-sum* e *bounded fast-max-sum*, os quais foram desenvolvidos com base no algoritmo soma-máxima. Além desse trabalho, outros algoritmos para alocação de tarefas foram desenvolvidos também com base em DCOPs, por exemplo, o LA-DCOP [Scerri et al. 2005] e Swarm-GAP [Ferreira Jr et al. 2008].

## V. Trabalhos Futuros

Pretende-se investigar domínios de problemas reais no qual a alocação de tarefas/recursos seja carente de abordagens distribuídas e eficientes. A partir disso, um determinado domínio será escolhido e modelado sob a perspectiva de sistemas multiagentes, adotando DCOP para realização da coordenação. Além disso, diferentes algoritmos serão avaliados a fim de encontrar o mais adequado para o domínio em questão.

## References


[Apt 2003] Apt, K. (2003). *Principles of constraint programming*. Cambridge University Press.

[Farinelli et al. 2008] Farinelli, A., Rogers, A., Petcu, A., e Jennings, N. R. (2008). Decentralised coordination of low-power embedded devices using the max-sum algorithm. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems - Volume 2*, AAMAS '08, pages 639–646, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

[Ferreira Jr et al. 2008] Ferreira Jr, P. R., Boffo, F. S., e Bazzan, A. L. (2008). Using swarm-gap for distributed task allocation in complex scenarios. In *Massively Multi-Agent Technology*, pages 107–121. Springer.

[Kschischang et al. 2001] Kschischang, F., Member, S., Frey, B. J., e andrea Loeliger, H. (2001). Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47:498–519.

[Macarthur 2011] Macarthur, K. S. (2011). *Multi-Agent Coordination for Dynamic Decentralised Task Allocation*. PhD thesis, University of Southampton.

[Mailler e Lesser 2004] Mailler, R. e Lesser, V. (2004). Solving distributed constraint optimization problems using cooperative mediation. In *International Conference on Autonomous Agents: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-*, volume 1, pages 438–445.

[Modi et al. 2005] Modi, P. J., Shen, W.-M., Tambe, M., e Yokoo, M. (2005). Adopt: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161(1):149–180.

[Petcu e Faltings 2005] Petcu, A. e Faltings, B. (2005). A scalable method for multiagent constraint optimization. In *International Joint Conference on Artificial Intelligence*, volume 19, page 266. LAWRENCE ERLBAUM ASSOCIATES LTD.

[Rogers et al. 2011] Rogers, A., Farinelli, A., Stranders, R., e Jennings, N. R. (2011). Bounded approximate decentralised coordination via the max-sum algorithm. *Artificial Intelligence*, 175(2):730–759.

[Scerri et al. 2005] Scerri, P., Farinelli, A., Okamoto, S., e Tambe, M. (2005). Allocating tasks in extreme teams. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 727–734. ACM.

[Schiex et al. 1995] Schiex, T., Fargier, H., Verfaillie, G., et al. (1995). Valued constraint satisfaction problems: Hard and easy problems. In *International Joint Conference on Artificial Intelligence*, volume 14, pages 631–639. Citeseer.

[Stranders et al. 2009] Stranders, R., Farinelli, A., Rogers, A., e Jennings, N. R. (2009). Decentralised coordination of mobile sensors using the max-sum algorithm. In *Proceedings of the 21st international jont conference on Artifical intelligence*, pages 299–304. Morgan Kaufmann Publishers Inc.

[Yeoh 2010] Yeoh, W. (2010). *SPEEDING UP DISTRIBUTED CONSTRAINT OPTIMIZATION SEARCH ALGORITHMS*. PhD thesis, UNIVERSITY OF SOUTHERN CALIFORNIA.

[Yokoo et al. 1992] Yokoo, M., Ishida, T., Durfee, E. H., e Kuwabara, K. (1992). Distributed constraint satisfaction for formalizing distributed problem solving. In *Distributed Computing Systems, 1992., Proceedings of the 12th International Conference on*, pages 614–621. IEEE.