Times Assíncronos:
Uma nova técnica para o *Flow Shop Problem*

Hélvio Pereira Peixoto
Pedro Sérgio de Souza

Relatório Técnico DCC–94-06

Julho de 1994

Times Assíncronos: Uma nova técnica para o *Flow Shop Problem*

Hélvio Pereira Peixoto Pedro Sérgio de Souza

Universidade Estadual de Campinas
Departamento de Ciência da Computação - IMECC
Cx.Postal - 6065 CEP - 13081.970
Campinas - SP

Sumário

Este artigo apresenta a descrição de *A-Teams* para o problema de escalonamento de tarefas denominado *Flow Shop Problem*. *A-Teams* perfazem uma nova técnica de resolução de problemas baseada na utilização simultânea de diversos algoritmos heurísticos, que cooperam entre si de maneira sinérgica, encontrando soluções ótimas ou quase ótimas, as quais não seriam encontradas pelos algoritmos isoladamente. Os resultados obtidos são comparados com os resultados fornecidos pelas heurísticas mais significativas encontradas na literatura.

Abstract

The purpose of this paper is to describe a new method for solving the Flow Shop Scheduling Problem. This method, named Asynchronous Teams (A-Teams), uses simultaneously a "team" of heuristics algorithms that help each other in finding optimal or near optimal solutions, which would not be achieved if the algorithms were used isolately. The results of several A-Teams are compared with those from other well-known heuristics.

1 Introdução

Após o advento da teoria que trata dos problemas NP-Completos e NP-Difíceis, muitos problemas que pertencem a estas classes foram contemplados por dezenas de algoritmos heurísticos, que se propõem a resolvê-los aproximadamente, uma vez que algoritmos exatos polinomiais são improváveis que existam [GJ79]. Para ratificar tal fato, basta verificar o número de trabalhos publicados onde os problemas combinatórios são resolvidos heuristicamente. Como exemplo, temos o problema do Caixeiro Viajante [LLKS85], o problema de Roteamento de Veículos [Lap92], o problema de escalonamento de tarefas *Flow Shop Problem* e *Job Shop Problem* [ML93], entre outros.

Uma vez identificado o problema em questão e as respectivas heurísticas disponíveis, qual delas é a mais adequada? Esta pergunta pode levar a uma análise exaustiva de todas elas, tomando como parâmetro de comparação, a complexidade de implementação, qualidade das soluções, eficiência computacional, robustez, e muitas outras características [ZE81, BM81]. Um estudo detalhado pode respaldar a escolha de uma delas, porém, ao utilizar apenas uma, despreza-se as eventuais qualidades das demais. Não haveria então uma maneira de organizar um "time" de heurísticas de modo a tirar proveito das virtudes de cada uma e obter soluções melhores que as obtidas por elas isoladamente? Já em 1975, Weiner sugeria a utilização de uma heurística de melhoria[1] sobre a solução gerada por uma heurística de construção[2] [Wei75].

Uma abordagem mais elaborada para organizar heurísticas pode ser encontrada em [Sou93]. Em seu trabalho, Souza propôs uma organização de heurísticas para o problema do Caixeiro Viajante, obtendo via de regra, soluções ótimas. Esta nova técnica, intitulada *Asynchronous Teams* (*A-Teams*), tem sido aplicada com sucesso a outros problemas combi-

[1] Heurísticas que partem de uma solução válida e as modificam à procura de soluções melhores.

[2] Heurísticas que a partir da descrição da instância do problema constrõem uma solução elemento a elemento.

natórios de grande porte [Mur92, Che92], e, por isso a motivação de aplicar *A-Teams* ao *Flow Shop Problem*.

O presente artigo descreve, nas próximas seções, o problema de escalonamento de tarefas *Flow Shop Problem*, as principais características dos *A-Teams*, os *A-Teams* propostos para o *Flow Shop Problem* e os resultados obtidos.

2 *Flow Shop Problem*

Flow Shop Problem (FSP), rotulado $n/m/F/F_{max}$ por Conway *et al.* [CMM67], é um problema combinatório clássico de escalonamento de *jobs*, e pode ser definido como segue: um conjunto de n *jobs* (ou itens, serviços, etc), cada *job* composto por m operações, deve ser processado na mesma seqüência nas m máquinas disponíveis, dado que o tempo de processamento t_{ij} do *job* i na máquina j é fixo, não negativo, e pode ser zero se o respectivo *job* não for processado pela referida máquina. O objetivo é encontrar uma seqüência de *jobs* que minimize o *Makespan*, ou seja, o tempo entre o início do primeiro *job* na primeira máquina e o término do último *job* na última máquina. É ainda considerado que:

- todo *job* deve ser processado no máximo uma vez em cada máquina;
- a operação i de qualquer *job* é processada pela máquina i;
- toda máquina é capaz de processar apenas um *job* a cada instante;
- todo *job* é processado no máximo em uma máquina a cada instante;
- os tempos de *setup* (tempo de configuração), caso existam, estão incluidos no tempo de processamento e são independentes da seqüência dos *jobs* e,
- as operações são indivisíveis.

Existem algoritmos eficientes que geram soluções ótimas em tempo polinomial para casos particulares do FSP [Bak74]. Estes casos consistem na restrição do número de máquinas (2 ou 3) ou FSP muito simples. Porém, o FSP geral é inerentemente complexo, a saber, NP-Difícil [GJS76].

Métodos exatos como enumeração explícita e *branch and bound* não são adequados na prática, pois, no pior caso, todas as $n!$ possíveis soluções do problema original são verificadas.

Para as instâncias de tamanho real onde as soluções devem ser conhecidas XSem um tempo determinado (geralmente pequeno), as heurísticas são mais úteis. Algumas heurísticas polinomiais da literatura e duas novas, propostas por estes autores, estão listadas abaixo:

- Heurísticas de Page [Pag61]: Page desenvolveu algumas heurísticas baseando-se em procedimentos de ordenação. *Merging* (M) consiste em ordenar um conjunto de *jobs* usando a abordagem do algoritmo de *MergSort* [CLR90], isto é, há particionamentos e realocações similarmente ao *MergSort*, ordenando pela ordem que minimize o Makespan. *Individual Exchange* (IE) se baseia no algoritmo *Buble-Sort* [CLR90] e finalmente o *Group Exchange*, que permuta grupos de *jobs* ao invés de *jobs* isolados, tal como o *Individual Exchange*.

- Heurística de Palmer [Pal65] (Palmer): Palmer propôs um algoritmo que escalona primeiro os *jobs* de maior prioridade, ou seja, os *jobs* cujos tempos de processamento em cada máquina tendem a crescer.

- Heurística de Gupta [Gup71] (Gupta): Gupta sugeriu um algoritmo similar ao de Palmer, exceto pelo fato das prioridades serem definidas pela generalização da regra utilizada por Johnson, para o FSP de três máquinas [Bak74].

- Heurística de Campbell, Dudek e Smith [CDS70] (CDS): Campbell *et al.* também propuseram um algoritmo que consite na genera-

lização do algoritmo de Johnson para três máquinas [Bak74]. Johnson converte o problema de três máquinas em um novo problema de duas máquinas e então utiliza o seu conhecido algoritmo de Johnson [Bak74] para resolvê-lo. O método proposto por Campbell *et al.* cria $(m-1)$ novos sub-problemas de duas máquinas, resolve-os pelo algoritmo de Johnson e a seqüência de *jobs* que produzir o menor *Makespan* é escolhida como solução para o problema original.

- Heurística de Nawaz, Enscore e Ham [NEH83] (NEH): Nawaz *et al.* desenvolveram uma heurística onde os dois *jobs* de maior tempo total de processamento são escalonados primeiro, como se somente os dois estivessem disponíveis. Então, em ordem não crescente de tempo total de processamento, insere-se, um a um, os *jobs* nas posições possíveis e escolhe-se a posição que minimiza *Makespan*.

- Heurística de melhoria NEH (MNEH): Estes autores propuseram uma heurística de melhoria baseada na heurística NEH. Dada uma solução, retira-se o *job* de maior tempo total de processamento e verifica-se em qual das $(n-1)$ posições este *job* deve ser inserido, de modo a minimizar o *Makespan*. Esse procedimento é efetuado para os n *jobs*, conforme o tempo total de processamento de cada um.

- Heurística Windows: Estes autores propuseram uma heurística semelhante à proposta por Storer *et al.* [SWV92]. Nesta nova heurística considera-se que as soluções geradas pelas de Palmer, CDS, Gupta e NEH são regras de prioridades. Escalona-se então os *jobs* conforme o seguinte procedimento: sorteie uma das regras anteriores e escalone os $n/10$ primeiros *jobs* conforme a regra sorteada. Para cada lote subsequente de $n/10$ *jobs* é repetido o mesmo procedimento. A heurística termina quando todos os lotes estiverem escalonados.

Outras heurísticas podem ser encontradas na literatura [Dan77, KS80, SS82, PPE84, HR88, WH89, Tai90, OS90]. Taillard [Tai90], Widmer

et al. [WH89] e Park *et al.* [PPE84] apontam NEH como a melhor heurística polinomial determinística.

A próxima seção descreve as principais características dos *A-Teams*.

3 Caracterização dos *A-Teams*

Ao dispormos de uma coleção de heurísticas que se propõem a resolver um problema, pode-se combiná-las com o objetivo de obter soluções melhores que as geradas por elas isoladamente. Por exemplo: Sejam A, B, C e D quatro heurísticas determinísticas distintas de um dado problema P tal que A, B e C são heurísticas de melhoria e D uma heurística de construção. Pode-se combiná-las de seis maneiras distintas como ilustrado na figura 1.

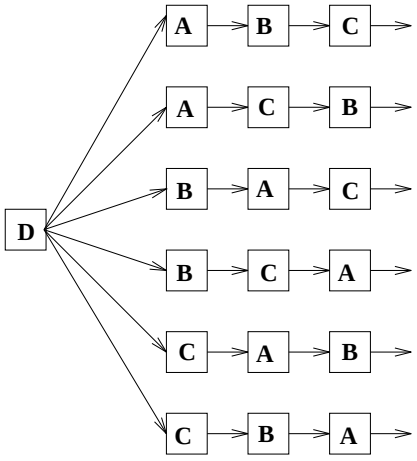

Figura 1: Seqüências possíveis para heurísticas A, B, e C de melhoria e D de construção.

Qual dessas seqüências produz os melhores resultados? Seja qual for a resposta, têm-se dois inconvenientes principais:

1. A seqüência escolhida pode gerar soluções de boa qualidade apenas para um subconjunto das instâncias de P.

2. Mesmo não se levando em consideração o item anterior, o número de combinações possíveis pode-se tornar proibitiva a tentativa de analisar qual seqüência seria a "melhor".

Dado tais dificuldades, seria interessante a existência de uma organização de software que explorasse a interação entre as heurísticas inteligentemente.

A-Teams é uma nova técnica de resolução de problemas baseada na utilização simultânea e assíncrona de diversos agentes (algoritmos heurísticos), que cooperam entre si de maneira sinérgica, obtendo soluções ótimas ou quase ótimas, as quais não seriam encontradas pelos agentes isoladamente [Sou93, ST91].

Um *A-Team* é composto por uma ou mais memórias compartilhadas, onde os agentes podem ler e escrever dados (soluções), formando uma organização [Tal93, ST93]. Os dados são lidos conforme a política de leitura de cada agente, que é livre para escolher o dado que melhor lhe aprouver. Como as memórias compartilhadas não podem incorporar dados indefinidamente (as memórias são finitas), a cada uma está associado um agente denominado *Destruidor*. As únicas funções do *Destruidor* são julgar e eliminar, baseado em uma política de destruição, qual dado deve ser eliminado para abrir espaço a um novo. A política de destruição é geralmente relacionada à qualidade dos dados.

Um exemplo de representação gráfica de *A-Teams* é dada na figura 2 onde três memórias são compartilhadas entre nove agentes. Os retângulos representam as memórias e as setas os agentes. O sentido das setas indica donde os agentes retiram os dados e aonde os agentes depositam os seus resultados. Por exemplo, o agente A lê e escreve na memória M1, já o agente H lê de M3 e escreve em M2. Os caminhos gerados pelas setas nesta representação definem o fluxo de dados do *A-Team*.

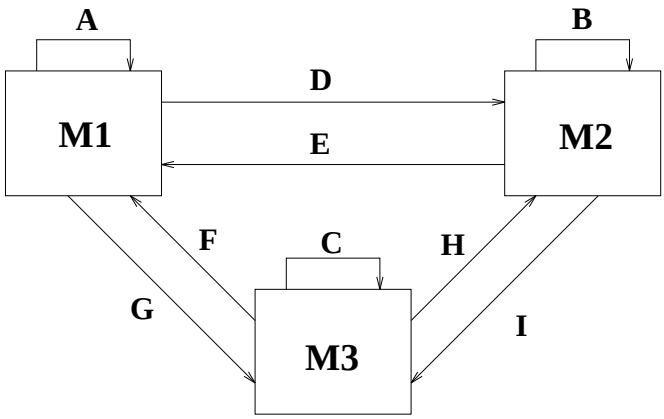

Figura 2: Um exemplo de *A-Teams* com três memórias e nove agentes.

As propriedades de uma organização que caracterizam um *A-Team* são:

Agentes Autônomos : Um agente é considerado autônomo se nenhum outro agente interfere na escolha de seus dados de entrada, quando e como processá-los. Uma vez que os agentes são autônomos, novos agentes podem entrar e sair da organização sem qualquer notificação prévia ou supervisão dos demais.

Fluxo de Dados Cíclico : O fluxo de dados cíclico permite que as mudanças realizadas por um agente num dado sejam "acessíveis" aos demais agentes, permitindo assim *feedback* e interação entre eles. Isto gera a possibilidade de se ter modificações e alterações nos dados de forma a criar outros novos e melhores para o problema em questão.

Comunicação Assíncrona : Agentes podem ler e escrever nas memórias sempre que lhes aprouver, livres de qualquer sincronismo, o que permite que sejam executados concorrente ou paralelamente.

Outra característica dos *A-Teams* é o fato do tempo de comunição entre seus agentes ser insignificante frente ao tempo de processamento entre duas comunicações sucessivas (granularidade grossa dos agentes) para problemas suficientemente grandes. Essas duas últimas características (assincronismo e granularidade grossa) aliadas à autonomia dos agentes tornam os *A-Teams* adequados a serem executados em sistemas computacionais distribuídos e paralelos.

A próxima seção descreve alguns *A-Teams* para o *Flow Shop Problem*.

4 *A-Teams* para o *Flow Shop Problem*

Ao analisar as soluções fornecidas pelas heurísticas mencionadas anteriormente, observa-se que vários *jobs* ocupam as mesmas posições relativas em cada solução. Portanto, existe um consenso entre as heurísticas de que alguns *jobs* devam ser escalonados primeiro do que outros. Usando este fato, os autores deste artigo propuseram um *A-Team* para o FSP cuja topologia está representada na figura 3 (a).

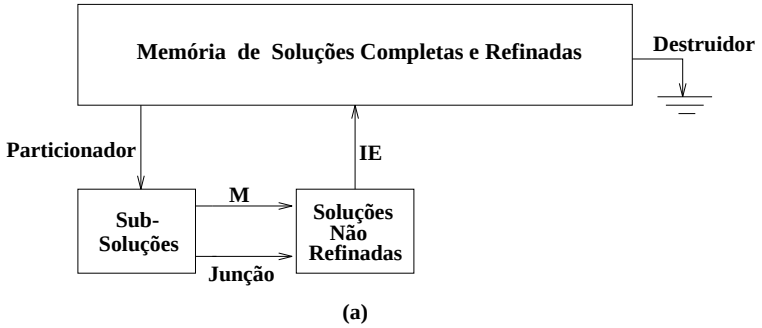

Figura 3: *A-Teams* para o FSP

O algoritmo *Particionador* mostrado na figura 3 (a) consulta quatro soluções na memória de soluções completas e refinadas, e verifica para cada *job* quantas vezes ele precede os demais em cada solução. O *Partici-*

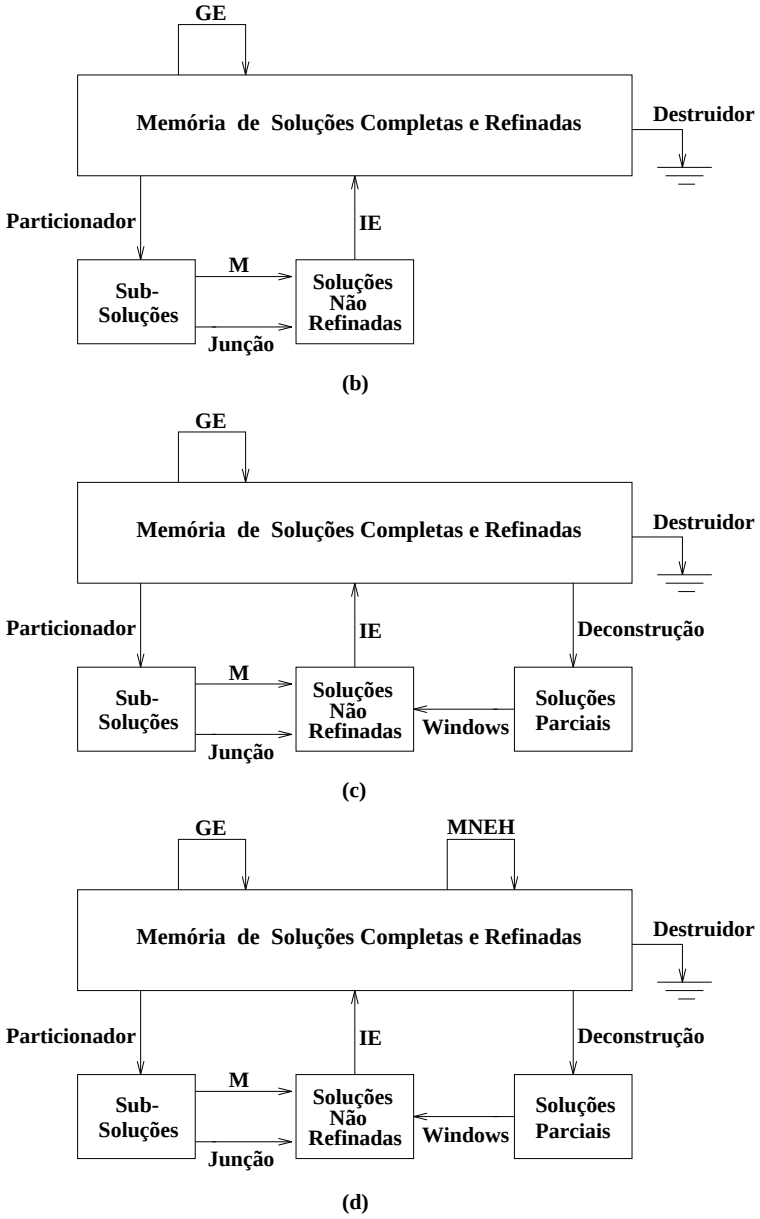

Figura 3: *A-Teams* para o FSP

onador então cria duas listas de $(n/2)$ elementos tal que a primeira lista contém os *jobs* mais "precedidos" e a segunda lista os demais. Em qualquer uma das duas listas criadas, se o *job* i precede o *job* j, então o *job* i possui um número maior ou igual de precedências nas quatro amostras em relação ao *job* j e, por conseqüência, o último *job* da primeira lista possui "precedência" maior ou igual ao primeiro *job* da segunda lista. As duas listas (ou sub-soluções) são armazenadas aos pares na memória de sub-soluções. Estas sub-soluções são retiradas pelos algoritmos *M* (*Merge*, descrito anteriormente) e *Junção*. A única função do algoritmo *Junção* é unir o final da primeira lista ao início da segunda. As soluções desses dois algoritmos, que são válidas, são armazenadas na memória de soluções não refinadas que, através da heurística de melhoria *IE*, são aprimoradas e depositadas na memória de soluções completas e refinadas. O algoritmo *Destruidor* descarta qualquer solução proposta cujo *Makespan* seja maior que o *Makespan* da pior solução da memória de soluções completas. Quando uma solução "boa" é proposta à memória de soluções completas e não há espaço disponível, o *Destruidor* elimina a pior solução.

Este *A-Team* "básico" da figura 3 (a) pode evoluir através da incorporação de outras heurísticas. Uma evolução possível é a inserção da heurística de melhoria *GE*, um novo ciclo contendo a heurística *Windows* e por fim a heurística *MNEH*. Os *A-Teams* resultantes estão ilustrados na figura 3 (b) (c) e (d), respectivamente.

O algoritmo *Deconstrutor* consulta duas soluções na memória de soluções completas e refinadas, verifica quais as posições que possuem *jobs* em comum e então cria uma nova solução parcial com esses *jobs* nas respectivas posições. Estas soluções são armazenadas na memória de soluções parciais. A heurística de construção Windows escalona em cada solução parcial os *jobs* ainda não escalonados e escreve a solução obtida na memória de soluções não refinadas. As heurísticas de melhoria *GE* e *MNEH* lêem e escrevem suas soluções na memória se soluções completas. A heurística *GE* escolhe aleatoriamente uma solução para leitura enquanto a *MNEH* lê a melhor solução. Tanto a *GE* como a *MNEH*

procuram outras soluções para leitura caso a solução escolhida já tenha sido processada por elas anteriormente.

5 Resultados

Os testes foram efetuados em instâncias de quatro tamanhos geradas conforme o algoritmo[3] proposto por Taillard [Tai93]. Essas instâncias foram consideradas "difíceis" por Taillard que usou *Tabu Search* com número de iterações na ordem de 10^4, obtendo assim a melhor solução conhecida para cada uma.

Os resultados obtidos pelo *A-Team* da figura 3 (d) estão na tabela 1. As três primeiras colunas indicam os valores obtidos pelas heurísticas *Palmer, CDS* e *NEH*. Para cada instância executou-se cinco vezes o *A-Team* da figura 3 (d) e a quarta coluna indica o melhor resultado e a média obtidas nas cinco execuções. A última coluna contém o melhor *Makespan* conhecido (MMC) fornecido por Taillard para cada instância. Em todos os testes 60% da memória de soluções completas e refinadas foi iniciada com soluções geradas aleatoriamente.

A tabela 2 indica o desempenho das heurísticas *Palmer, CDS, NEH* e do *A-Team* em relação aos melhores *Makespans* conhecidos. O *A-Team* quando não obteve o melhor *Makespan* conhecido esteve muito próximo.

Outra característica de *A-Teams* que pode ser observada da evolução do *A-Team* (a) da figura 3 ao (d) está nos resultados por eles obtidos. A medida que novos agentes (algoritmos) são incorporados na organização não só a qualidade das soluções geradas é progressivamente melhorada, como o tempo de processamento para obtê-las é sensivelmente reduzido (veja figura 4). Talukdar e Souza [TS92] denominam uma organização de *Eficiente em Escala* se a introdução de novos membros nesta organização melhora monotonicamente alguma medida de performance. A figura 4

[3] As sementes usadas para gerar as instâncias 20x5, 20x10, 50x10 e 100x10 são, respectivamente, 873654221, 587595453, 1958948863 e 1539989115.

	Palmer	CDS	NEH	*A-Team*		MMC
				Média	Melhor	
20 x 5	1384	1399	1286	1278	1278	1278
20x10	1917	1729	1680	1582	1582	1582
50x10	3422	3444	3136	3059	3046	3037
100x10	6143	6252	5860	5794	5781	5776

Tabela 1: Resultados obtidos pelas heurísticas Palmer, CDS, NEH e o *A-Team* da figura 3 (d) sobre as instâncias de n *jobs* x m máquinas.

	Palmer %	CDS %	NEH %	*A-Team*	
				Media %	Melhor %
20 x 5	8.9	9.47	0.63	0.0	0.0
20x10	21.0	9.29	6.19	0.0	0.0
50x10	13.0	13.0	3.26	0.72	0.30
100x10	6.35	8.24	1.45	0.31	0.087

Tabela 2: Percentagens relativas ao melhor *Makespan* conhecido das soluções obtidas pelas heurísticas de Palmer, CDS, NEH e o *A-Team* da figura 3 (d) sobre as instâncias de n *jobs* x m máquinas.

mostra os resultados obtidos pelos *A-Teams* (a), (b), (c) e (d) da figura 3 sobre a instância 100x10 e atesta que os mesmos possuem eficiência em escala.

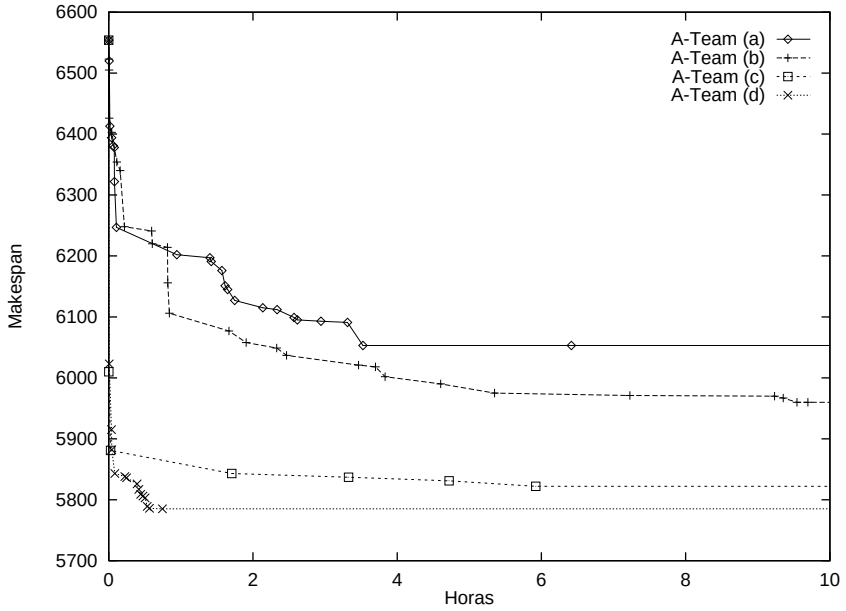

Figura 4: Performance dos *A-Team* (a), (b), (c) e (d) da figura 3 contra o tempo de processamento sobre a instância 100x10

A dinâmica da memória de soluções completas e refinadas pode ser compreendida através do gráfico da figura 5. O primeiro passo no *A-Team* consiste na iniciação de 60% da memória de soluções completas e refinadas com soluções geradas aleatoriamente. A partir do início de execução dos agentes a memória de soluções completas é progressivamente preenchida, e soluções com qualquer *Makespan* são aceitas afim de não limitar os possíveis caminhos iniciais de busca. Uma vez que a memória de soluções completas estiver completamente preenchida, o *Destruidor*, sempre que necessário, eliminará as piores soluções. Dessa maneira o *A-Team* "direciona" e "concentra" a busca em regiões do espaço

de soluções que, a princípio, contém as melhores soluções.

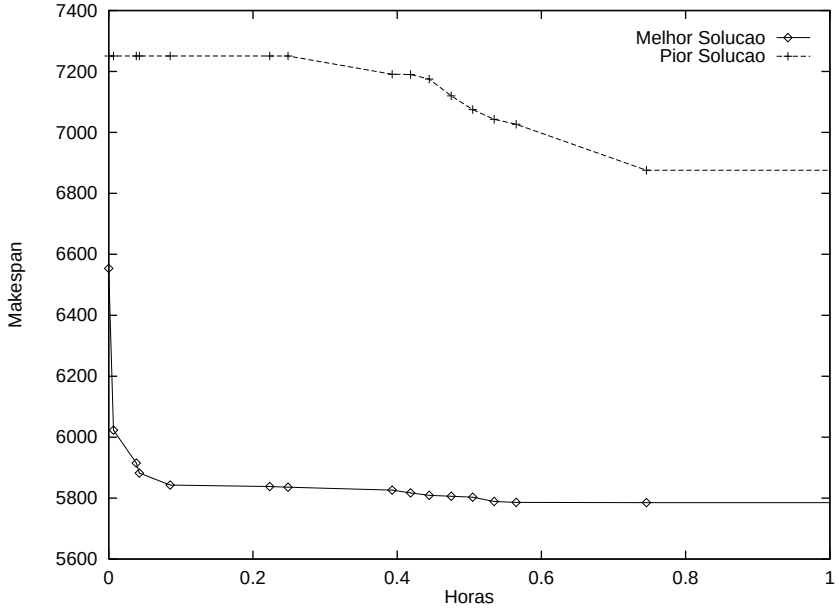

Figura 5: Dinâmica das soluções na memória de soluções completas e refinadas no decorrer do tempo.

Todos os testes foram efetuados em estações *SUN SPARCstation* 1+ e os tempos de processamento indicados no gráfico da figura 4 e 5 são tempos aproximados de CPU.

6 Conclusões

Mostramos que algoritmos heurísticos podem ser organizados de maneira a produzir soluções melhores que as produzidas por eles isoladamente, em particular para o problema de escalonamento de tarefas *Flow Shop Problem*. Esta nova técnica, intitulada *A-Teams*, é uma organização

de software que combina algoritmos heurísticos assíncrona e interativamente. Devido a autonomia, ao fato das comunicações entre os agentes serem assíncronas e a granularidade grossa dos algoritmos, os *A-Teams* são adequados a serem executados em sistemas paralelos e distribuídos. Mostramos também que os *A-Teams* para o *Flow Shop Problem* possuem eficiência em escala, ou seja, a eficiência da organização se mostra proporcional ao número de membros que a compõe.

Objetivando resultados ainda melhores, os autores deste artigo continuam trabalhando com novas topologias de *A-Teams* para o *Flow Shop Problem* e, futuramente, uma versão distribuída será apresentada.

Referências

[Bak74] K. R. Baker. *Introduction to Sequencing and Scheduling*. New York, John Wiley, 1974.

[BM81] M. Ball e M. Magazine. The design and analysis of heuristics. *Networks*, 11, 1981.

[CDS70] H. G. Campbell, R. A. Dudek, e M. L. Smith. An heuristic algorithm for n job m machine sequencing problem. *Managment Science*, 16(10), 1970.

[Che92] C. L. Chen. *Baysean Nets and A-Teams for Power System Fault Diagnosis*. PhD thesis, Electrical and Computer Engineering Department, Carnegie Mellow University, Pittsburgh, PA, 1992.

[CLR90] T. H. Cormem, C. E. Leiserson, e L. R. Rivest. *Introduction to algorithms*. McGraw-Hill, 1990.

[CMM67] R. W. Conway, W. L. Maxwell, e L. W. Miller. *Theory of scheduling*. Addison-Wesley, 1967.

[Dan77] D. G. Dannenbring. An evaluation of flow shop sequencing heuristics. *Managment Science*, 23(11), 1977.

[GJ79] R. M. Garey e D. S. Johnson. *Computers and intractability, a guide to the theory of NP-Completeness*. W.H. Freeman and co., 1979.

[GJS76] M. R. Garey, D. S. Johnson, e R. Sethi. The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*, 1(2), 1976.

[Gup71] J. N. D. Gupta. A functional heuristic algorithm for the flow shop scheduling problem. *Operational Research Quarterly*, 22(1), 1971.

[HR88] T. S. Hundal e J. Raigopal. An extension of palmer's heuristic for the flow shop scheduling problem. *International Journal of Production Research*, 26(6), 1988.

[KS80] J. R. King e S. Spachuis. Heuristics for flow-shop scheduling. *International Journal of Production Research*, 18(3), 1980.

[Lap92] G. Laporte. The vehicle routing problem: An orveview of exact and aproximate algorithms. *European Jornal on Operational Research*, 59, 1992.

[LLKS85] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnoy Kan, e D. B. Shimoys. *The Traveling Salesman Problem*. John Wiley, 1985.

[ML93] B. L. Maccarthy e Jiyin Liu. Addressing the gap in scheduling research: a review of optimization and heuristic methods in production scheduling. *International Journal of Production Research*, 31(1), 1993.

[Mur92] S. Murthy. *Synergy in cooperating agents: designing manipulators from task specifications*. PhD thesis, Electrical and Computer Engineering Department, Carnegie Mellow University, Pittsburgh, PA, 1992.

[NEH83] M. Nawaz, E. Enscore, e I. Ham. A heuristic algorithm for the m-machine/n-job flow-shop sequencing problem. *OMEGA, The International Journal of the Managment Science*, 11(1), 1983.

[OS90] F. A. Ogbu e D. K. Smith. The application of the simulated annealing algorithm to the solution of n/m/cmax flowshop problem. *Computers Opns. Res.*, 17(3), 1990.

[Pag61] E. S. Page. An approach to scheduling jobs on machines. *Journal of the Royal Statistical Society, Series B*, 23, 1961.

[Pal65] D. S. Palmer. Sequencing jobs through a multi-stage process in the minimum total time - a quick method fo obtaining a near optimum. *Operations Research Quarterly*, 16, 1965.

[PPE84] Y. B. Park, C. D. Pegden, e E. E. Enscore. A survey and evaluation of static flowshop scheduling heuristics. *International Journal of Production Research*, 22(1), 1984.

[Sou93] P. S. Souza. *Asyncronous Organizations for Multi-algorithm Problems*. PhD thesis, Electrical and Computer Engineering Department, Carnegie Mellon University, Pittsburg,PA, 1993.

[SS82] J. P. Stinson e A. W. Smith. A heuristic programming procedure for sequencing the static flow shop. *International Journal of Production Research*, 20(6), 1982.

[ST91] P. S. Souza e S. N. Talukdar. Genetic algorithms in asyncronous teams. *Proceedings of the Fourth International Conference on Genetic Algorithms, Morgan Kaufmann, Los Altos, CA*, 1991.

[ST93] P. S. Souza e S. N. Talukdar. Asyncronous organizations for multi-algorithm problems. *ACM Symposium on Applied Computing, Indianapolis, IN*, fevereiro de 1993.

[SWV92] Robert H. Storer, S. David Wu, e Renzo Vaccari. New search spaces for sequencing problems with application to job shop scheduling. *Management Science*, 38(10), 1992.

[Tai90] E. Taillard. Some efficient heuristic methods for the flow shop sequencing problem. *European Journal of Operational Research*, 47, 1990.

[Tai93] E. Taillard. Benchmarks for basic scheduling problems. *European Journal of Operational Research*, 64, 1993.

[Tal93] S. N. Talukdar. Asynchronous teams. *Fourth Symposium on Expert Systems Application to Power Systems, Melbourn, Australia*, 1993.

[TS92] S. N. Talukdar e P. S. Souza. Scale efficient organizations. *Proceedings of the 1992 IEEE International Conference on Systems, Man and Cybernetics, Chicago*, outubro de 1992.

[Wei75] P. Weiner. Heuristics. *Networks*, 5, 1975.

[WH89] M. Widmer e A. Hertz. A new heuristic method for the flow shop sequencing problem. *European Journal of Operational Research*, 41, 1989.

[ZE81] S. H. Zanakis e J. R. Evans. Heuristic "optimization": why, when, and how to use it. *Interfaces*, 11(5), 1981.

Relatórios Técnicos – 1992

92-01 **Applications of Finite Automata Representing Large Vocabularies,** *C. L. Lucchesi, T. Kowaltowski*

92-02 **Point Set Pattern Matching in d-Dimensions,** *P. J. de Rezende, D. T. Lee*

92-03 **On the Irrelevance of Edge Orientations on the Acyclic Directed Two Disjoint Paths Problem,** *C. L. Lucchesi, M. C. M. T. Giglio*

92-04 **A Note on Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams,** *W. Jacometti*

92-05 **An (l, u)-Transversal Theorem for Bipartite Graphs,** *C. L. Lucchesi, D. H. Younger*

92-06 **Implementing Integrity Control in Active Databases,** *C. B. Medeiros, M. J. Andrade*

92-07 **New Experimental Results For Bipartite Matching,** *J. C. Setubal*

92-08 **Maintaining Integrity Constraints across Versions in a Database,** *C. B. Medeiros, G. Jomier, W. Cellary*

92-09 **On Clique-Complete Graphs,** *C. L. Lucchesi, C. P. Mello, J. L. Szwarcfiter*

92-10 **Examples of Informal but Rigorous Correctness Proofs for Tree Traversing Algorithms,** *T. Kowaltowski*

92-11 **Debugging Aids for Statechart-Based Systems,** *V. G. S. Elias, H. Liesenberg*

92-12 **Browsing and Querying in Object-Oriented Databases,** *J. L. de Oliveira, R. de O. Anido*

Relatórios Técnicos – 1993

93-01 **Transforming Statecharts into Reactive Systems,** *Antonio G. Figueiredo Filho, Hans K. E. Liesenberg*

93-02 **The Hierarchical Ring Protocol: An Efficient Scheme for Reading Replicated Data,** *Nabor das C. Mendonça, Ricardo de O. Anido*

93-03 **Matching Algorithms for Bipartite Graphs,** *Herbert A. Baier Saip, Cláudio L. Lucchesi*

93-04 **A lexBFS Algorithm for Proper Interval Graph Recognition,** *Celina M. H. de Figueiredo, João Meidanis, Célia P. de Mello*

93-05 **Sistema Gerenciador de Processamento Cooperativo,** *Ivonne. M. Carrazana, Nelson. C. Machado, Célio. C. Guimarães*

93-06 **Implementação de um Banco de Dados Relacional Dotado de uma Interface Cooperativa,** *Nascif A. Abousalh Neto, Ariadne M. B. R. Carvalho*

93-07 **Estadogramas no Desenvolvimento de Interfaces,** *Fábio N. de Lucena, Hans K. E. Liesenberg*

93-08 **Introspection and Projection in Reasoning about Other Agents,** *Jacques Wainer*

93-09 **Codificação de Seqüências de Imagens com Quantização Vetorial,** *Carlos Antonio Reinaldo Costa, Paulo Lício de Geus*

93-10 **Minimização do Consumo de Energia em um Sistema para Aquisição de Dados Controlado por Microcomputador,** *Paulo Cesar Centoducatte, Nelson Castro Machado*

93-11 **An Implementation Structure for RM-OSI/ISO Transaction Processing Application Contexts,** *Flávio Morais de Assis Silva, Edmundo Roberto Mauro Madeira*

93-12 **Boole's conditions of possible experience and reasoning under uncertainty,** *Pierre Hansen, Brigitte Jaumard, Marcus Poggi de Aragão*

93-13 **Modelling Geographic Information Systems using an Object Oriented Framework,** *Fatima Pires, Claudia Bauzer Medeiros, Ardemiris Barros Silva*

93-14 **Managing Time in Object-Oriented Databases,** *Lincoln M. Oliveira, Claudia Bauzer Medeiros*

93-15 **Using Extended Hierarchical Quorum Consensus to Control Replicated Data: from Traditional Voting to Logical Structures,** *Nabor das Chagas Mendonça, Ricardo de Oliveira Anido*

93-16 $\mathcal{LL}$ **– An Object Oriented Library Language Reference Manual,** *Tomasz Kowaltowski, Evandro Bacarin*

93-17 **Metodologias para Conversão de Esquemas em Sistemas de Bancos de Dados Heterogêneos,** *Ronaldo Lopes de Oliveira, Geovane Cayres Magalhães*

93-18 **Rule Application in GIS – a Case Study,** *Claudia Bauzer Medeiros, Geovane Cayres Magalhães*

93-19 **Modelamento, Simulação e Síntese com VHDL,** *Carlos Geraldo Krüger e Mário Lúcio Côrtes*

93-20 **Reflections on Using Statecharts to Capture Human-Computer Interface Behaviour,** *Fábio Nogueira de Lucena e Hans Liesenberg*

93-21 **Applications of Finite Automata in Debugging Natural Language Vocabularies,** *Tomasz Kowaltowski, Cláudio Leonardo Lucchesi e Jorge Stolfi*

93-22 **Minimization of Binary Automata,** *Tomasz Kowaltowski, Cláudio Leonardo Lucchesi e Jorge Stolfi*

93-23 **Rethinking the DNA Fragment Assembly Problem,** *João Meidanis*

93-24 EGOLib — **Uma Biblioteca Orientada a Objetos Gráficos,** *Eduardo Aguiar Patrocínio, Pedro Jussieu de Rezende*

93-25 **Compreensão de Algoritmos através de Ambientes Dedicados a Animação,** *Rackel Valadares Amorim, Pedro Jussieu de Rezende*

93-26 GeoLab: **An Environment for Development of Algorithms in Computational Geometry,** *Pedro Jussieu de Rezende, Welson R. Jacometti*

93-27 **A Unified Characterization of Chordal, Interval, Indifference and Other Classes of Graphs,** *João Meidanis*

93-28 **Programming Dialogue Control of User Interfaces Using Statecharts,** *Fábio Nogueira de Lucena e Hans Liesenberg*

93-29 EGOLib – **Manual de Referência,** *Eduardo Aguiar Patrocínio e Pedro Jussieu de Rezende*

Relatórios Técnicos – 1994

94-01 **A Statechart Engine to Support Implementations of Complex Behaviour,** *Fábio Nogueira de Lucena, Hans K. E. Liesenberg*

94-02 **Incorporação do Tempo em um SGBD Orientado a Objetos,** *Ângelo Roncalli Alencar Brayner, Claudia Bauzer Medeiros*

94-03 **O Algoritmo KMP através de Autômatos,** *Marcus Vinícius A. Andrade e Cláudio L. Lucchesi*

94-04 **On Edge-Colouring Indifference Graphs,** *Celina M. H. de Figueiredo, João Meidanis, Célia Picinin de Mello*

94-05 **Using Versions in GIS,** *Claudia Bauzer Medeiros and Geneviève Jomier*

Departamento de Ciência da Computação — IMECC
Caixa Postal 6065
Universidade Estadual de Campinas
13081-970 – Campinas – SP
BRASIL

reltec@dcc.unicamp.br